

- 合理、完善的知识体系结构
- 内容丰富，重点突出，应用性强
- 免费提供相关程序源代码下载
- 深入、详细剖析 MATLAB 工程应用技术

MATLAB/Simulink 建模与仿真实例精讲

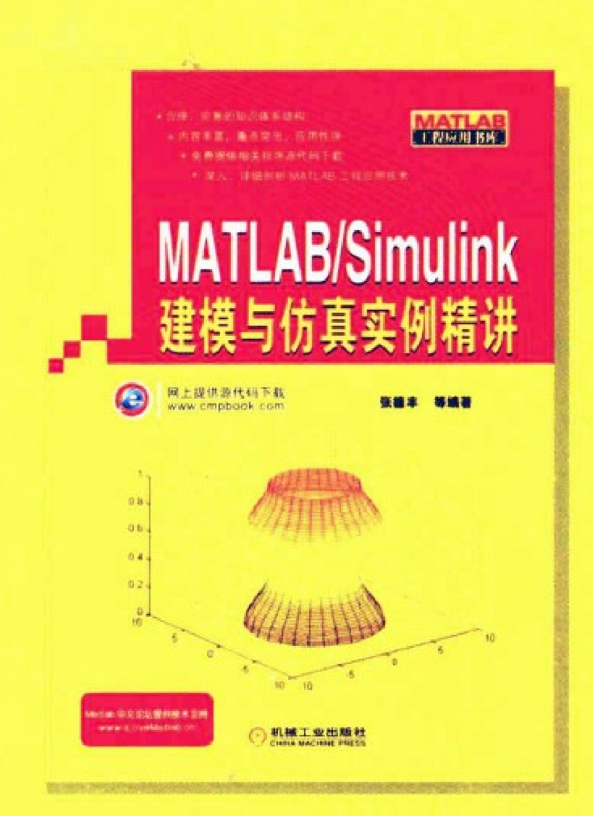


网上提供源代码下载
www.cmpbook.com

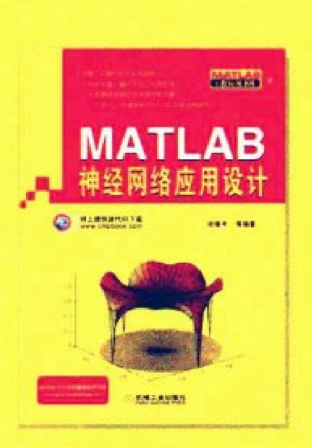
张德丰　等编著



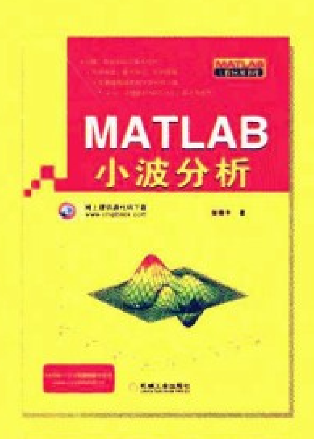
ISBN 978-7-111-29326-2
策　　划：丁 诚　吴鸣飞
封面设计：ZSC 子时文化 ZiShi Culture



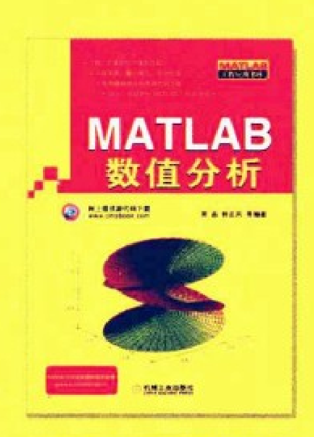
ISBN 978-7-111-29326-2
定价：46.00元



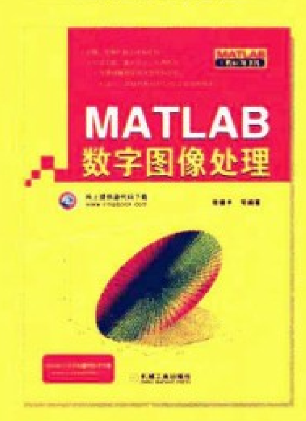
ISBN 978-7-111-25612-0
定价：39.00元



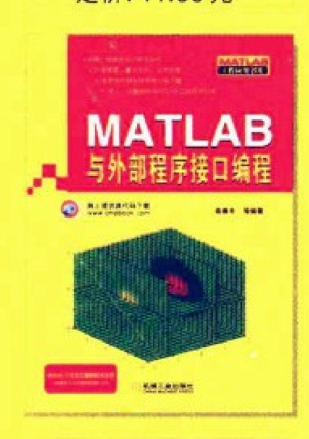
ISBN 978-7-111-25613-7
定价：41.00元



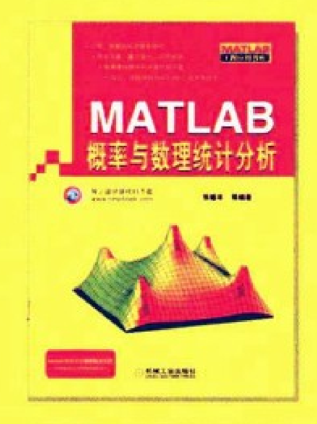
ISBN 978-7-111-25707-3
定价：39.00元



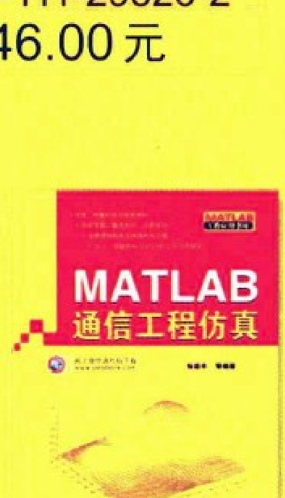
ISBN 978-7-111-25735-6
定价：39.00元



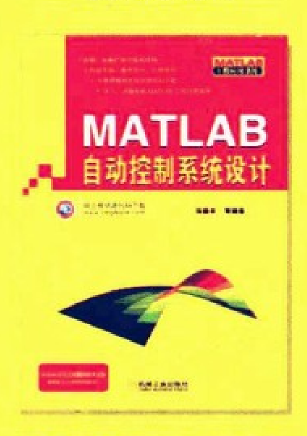
ISBN 978-7-111-25706-6
定价：42.00元



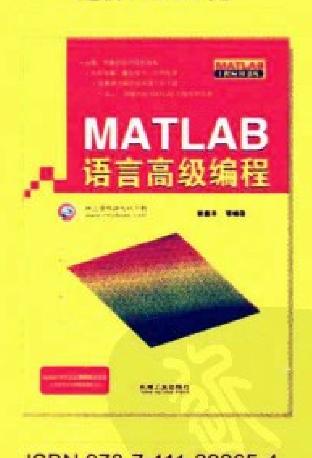
ISBN 978-7-111-29325-5
定价：41.00元



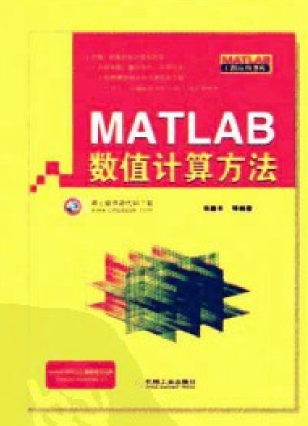
ISBN 978-7-111-29323-1
定价：46.00元



ISBN 978-7-111-29308-8
定价：45.00元



ISBN 978-7-111-29265-4
定价：45.00元



ISBN 978-7-111-29324-8
定价：42.00元

上架建议　计算机/辅助设计



地址：北京市百万庄大街22号　邮政编码：100037
电话服务　网络服务
社服务中心：(010)88361066　门户网：http://www.cmpbook.com
销 售 一 部：(010)68326294　教材网：http://www.cmpedu.com
销 售 二 部：(010)88379649
读者服务部：(010)68993821　封面无防伪标均为盗版

ISBN 978-7-111-29326-2
9 787111 293262 >

定价：46.00元

前　言

MATLAB 作为当前国际控制界最流行的面向工程与科学计算的高级语言，近年来得到了业界的一致认可，在控制系统的分析、仿真与设计方面得到了非常广泛的应用，其自身也因此得到了迅速的发展，功能不断扩充。本书大部分内容介绍了 MATLAB 在控制系统分析、仿真与设计中的应用。

另外，随着 MATLAB/Simulink 通信、信号处理专业函数库和专业工具箱的成熟，它们逐渐为广大通信技术领域的专家学者和工程师所熟悉，在通信理论研究、算法设计、系统设计、建模仿真和性能分析验证等方面的应用也更加广泛。Simulink 可视化仿真工具能够以非常直观的框图方式形象地对通信系统进行建模，并以“实时”和动画的方式来将模型仿真结果（如波形、频谱、数据曲线等）显示出来，更便于对通信系统的物理概念和运行过程的直观理解，所以近年来在通信工程专业中得到了广大师生的重视和广泛应用，在理论教学、课程实践环节，以及理论和技术前沿的研究中发挥了重要作用。对此，本书也进行了相关讲解。

全书共 9 章。第 1 章介绍了 MATLAB 及 Simulink 仿真基础知识，包括 MATLAB 简概、MATLAB R2009 的基本操作、仿真的一般过程与步骤等内容；第 2 章介绍了 MATLAB 的文件结构及其绘图介绍，包括 MATLAB 的程序结构、M 文件和基本图形绘制等内容；第 3 章介绍了 Simulink 仿真基础，包括 Simulink 操作概述、Simulink 模块处理分析、系统的仿真等内容；第 4 章介绍了 Simulink 建模与仿真高级应用，包括 Simulink 模块子系统和 S-函数建模与仿真等内容；第 5 章介绍了 Simulink 在控制系统中的应用，包括连续时间系统建模与仿真分析、离散系统建模与仿真分析等内容。第 6 章介绍了 Simulink 在电力系统的建模与仿真的应用，包括电力系统的模型分析、交直流调速系统的仿真分析等内容；第 7 章介绍了神经网络的仿真与分析，包括神经网络仿真的概述、Simulink 神经网络仿真示例等内容；第 8 章介绍了模糊逻辑控制的仿真分析，包括模糊逻辑控制概述、模糊逻辑控制的仿真分析应用示例等内容。第 9 章介绍了 Simulink 建模与仿真在通信系统中的应用，包括通信系统仿真的方法介绍、及 MATLAB/Simulink 在通信系统中的应用等内容。

本书可作为广大在校本科生和研究生的学习用书，也可以作为广大科研人员、学者、工程技术人员的参考用书。

参加本书编写的有张德丰、许华兴、王旭宝、王孟群、邓恒奋、卢国伟、卢焕斌、伍志聪、庄文华、庄浩杰、许业成、何沛彬、何佩贤、张水兰、张坚、李勇杰、李秋兰、李美妍、陈运英、陈景棠、梁家科、黄达中、陈楚明、林健锋、梁劲强、林振满、周品。

由于时间仓促，加之作者水平有限，所以错误和疏漏之处在所难免。在此，诚恳地期望得到各领域的专家和广大读者的批评指正。

作　者

全书共分 9 章。第 1 章介绍了 MATLAB 及 Simulink 仿真基础知识，包括 MATLAB 简介、MATLAB R2009 的基本操作、仿真的一般过程与步骤等内容；第 2 章介绍了 MATLAB 的文件结构及其绘图介绍，包括 MATLAB 的程序结构、M 文件和基本图形绘制等内容；第 3 章介绍了 Simulink 仿真基础，包括 Simulink 操作概述、Simulink 模块处理分析、系统的仿真等内容；第 4 章介绍了 Simulink 建模与仿真高级应用，包括 Simulink 模块子系统和 S-函数建模与仿真等内容；第 5 章介绍了 Simulink 在控制系统中的应用，包括连续时间系统建模与仿真分析、离散系统建模与仿真分析等内容；第 6 章介绍了 Simulink 在电力系统的建模与仿真的应用，包括电力系统的模型分析、交直流调速系统的仿真分析等内容；第 7 章介绍了神经网络的仿真与分析，包括神经网络仿真的概述、Simulink 神经网络仿真示例等内容；第 8 章介绍了模糊逻辑控制的仿真分析，包括模糊逻辑控制概述、模糊逻辑控制的仿真分析应用示例等内容；第 9 章介绍了 Simulink 建模与仿真在通信系统中的应用，包括通信系统仿真的方法介绍、及 MATLAB/Simulink 在通信系统中的应用等内容。

本书可作为广大在校本科生和研究生的学习用书，也可以作为广大科研人员、学者、工程技术人员的参考用书。

图书在版编目（CIP）数据

MATLAB/Simulink 建模与仿真实例精讲 / 张德丰等编著. —北京：机械工业出版社，2010.1

（MATLAB 工程应用书库）

ISBN 978-7-111-29326-2

Ⅰ. M… Ⅱ. 张… Ⅲ. ①计算机辅助计算－软件包，Matlab、Simulink－应用－系统建模②计算机辅助计算－软件包，Matlab、Simulink－应用－系统仿真 Ⅳ. TP391.9

中国版本图书馆 CIP 数据核字（2009）第 233728 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）

策划编辑：丁　诚　吴鸣飞

责任编辑：丁　诚　谷玉春

责任印制：洪汉军

三河市宏达印刷有限公司印刷

2010 年 1 月第 1 版 · 第 1 次印刷

184mm × 260mm · 26.25 印张 · 651 千字

0001 – 4000 册

标准书号：ISBN 978-7-111-29326-2

定价：46.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

社服务中心：（010）88361066

销 售 一 部：（010）68326294

销 售 二 部：（010）88379649

读者服务部：（010）68993821

网络服务

门户网：http://www.cmpbook.com

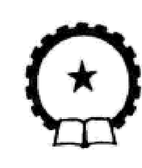
教材网：http://www.cmpedu.com

封面无防伪标均为盗版

MATLAB 工程

MATLAB/Simulink 建模与仿真实例精讲

张德丰　等编著



机 械 工 业 出 版 社

目　录









第 1 章　MATLAB 及 Simulink 仿真基础



1.1　MATLAB 概述

目前，在国际流行的科技应用软件中，数学类（区别于文字处理类和图像处理类）软件共有几十款之多。从它们的数学处理的原始内核来看，不外乎两种类型：数值计算型和数学分析型。前者如 MATLAB、Xmath 等，它们对大量数据具有较强的管理、计算和可视化能力，运行效率高；后者如 Mathematica、Maple 等，它们擅长于符号计算，可以得到问题的解析符号解和任意精度解，但处理大量数据时速度较慢。

1.1.1 MATLAB 发展史

MATLAB 是英文 MATrix LABoratory（矩阵实验室）的缩写。1980 年前后，时任美国墨西哥大学计算机科学系主任的 Cleve Moler 教授在给学习讲授线性数课程时，想教学生使用当时流行的线性代数软件包（Linpack）和基于特征值计算的软件包（Eispack），但发现用其他高级语言编程极为不便，于是 Cleve Moler 教授为学生编写了方便使用 Linpack 和 Eispack 的接口程序并命名为 MATLAB，这便是 MATLAB 的雏形。

早期的 MATLAB 是用 FORTRAN 语言编写的，尽管功能十分简单，但作为免费软件还是吸引了大批使用者。经过几年的校际流传，在 John Little 的推动下，由 John Little、Cleve Moler 和 Steve Bangert 合作，于 1984 年成立了 MathWorks 公司，并正式推出 MATLAB 第 1 版（DOS 版本）。从这时起，MATLAB 的核心采用 C 语言编写，功能越来越强大，除原有的数值计算功能外，还新增了图形处理功能。

以后，MATLAB 版本不断更新。MathWorks 公司于 1992 年推出了具有时代意义的 4.0 版本，并于 1993 年推出了其微型计算机版本，该版本可以在 Windows 3.X 上使用，使之应用范围越来越广。1994 年，推出了 4.2 版本扩充了 4.0 版本的功能，尤其在图形界面设计方面提供了新的方法。1997 年，MATLAB 5.0 版本问世，5.0 版本支持了更多的数据结构，如单元数据、结构数据、多维数组、对象与类等，使其成为一种更方便、更完善的编程语言。1999 年初推出的 MATLAB 5.3 版本在很多方面又进一步改进了 MATLAB 语言的功能，随之推出的全新版本的最优化工具箱和 Simulink 3.0 版本达到了很高水平。之后，MATLAB 还在不断改进和创新，2000 年 10 月，MATLAB 6.0 版本问世，在操作界面上有了很大改观，为用户的使用提供了很大方便；在计算性能方面，速度变得更快，性能也更好；在图形用户界面设计上更趋合理；与 C 语言接口及转换的兼容性更强；与之配套的 Simulink 4.0 版本的新功能也特别引人注目。2001 年 6 月推出的 MATLAB 6.1 版本及 Simulink 4.1 版本，功能已经十分强大。2002 年 6 月又推出了 6.5 版本及 Simulink 5.0 版本，在计算方法、图形功

能、用户界面设计、编程手段和工具等方面都有重大改进。

MATLAB R 系列是从 2006 年开始发布的，MathWorks 公司在技术层面上实现了一次飞跃。产品发布模式也将改变，将在每年的 3 月和 9 月进行两次产品发布，版本的命令方式为“R+年份+代码”，对应上下半年的代码分别是 a 和 b。每一次发布都会包含所有的产品模块，如产品的 new feature、bug fixes 和新产品模块的推出。MATLAB R2009a 是 MathWorks 公司 2009 年 3 月份推出的最新产品。

MathWorks 公司于 2008 年 11 月 7 日发布了 MATLAB R2009a。相比以前版本而言，MATLAB R2009a 不仅包括 MATLAB 和 Simulink 的新特性，同时还包含 81 个其他产品模块的升级和 Bug 修正。

目前，MATLAB 已经成为国际最流行的科学与工程计算软件之一。它以模块化的计算方法、可视化与智能化的人机交互功能、丰富的矩阵运算、图形绘制和数据处理函数，以及模块化图形的动态系统仿真工具 Simulink，成为控制系统设计和仿真领域最受欢迎的软件系统。

在欧美大学的应用代数、数理统计、自动控制、数字信号处理、模拟与数字通信、时间序列分析、动态系统仿真等课程的教科书，都把 MATLAB 作为其中的内容。在那里，MATLAB 是攻读学位的大学生、硕士生和博士生必须掌握的基本工具。

在国际学术界，MATLAB 已经被确认为准确、可靠的科学计算标准软件。在许多的国际学术刊物上（尤其是信息科学刊物），都可以看到 MATLAB 的应用。

在设计研究单位和工业部门，MATLAB 被认为是进行高效研究、开发的首选软件工具，如美国 National Instruments 公司信号测量分析软件 LabVIEW，Cadence 公司信号和通信分析设计软件 SPW 等，都是以 MATLAB 为主要支撑的。

1.1.2 MATLAB 的功能与特点

MATLAB 是一种高精度的科学计算语言，它将计算、可视化和编程结合在一个容易使用的环境中，在这个环境中，用户可以把提出的问题和解决问题的办法用熟悉的数学符号表示出来，它的典型使用包括：

- 数学和计算。
- 运算法则。
- 建模和仿真。
- 数据分析、研究和可视化。
- 科学的工程图形。
- 应用程序开发，包括创建图形用户接口。

MATLAB 是一个交互式系统，它的基本数据单元是数组，这个数组不要求固定的大小，因此可以让用户解决许多技术上的计算问题，特别是那些包含矩阵和矢量运算的问题。MATLAB 的命令表达与数学、工程中常用的习惯形式十分相似，与 C、FORTRAN 等高级语言相比，MATLAB 的语法规则更简单、表达更符合工程习惯。正因为如此，人们用 MATLAB 语言编写程序就有如在便笺上书写公式和求解，因而 MATLAB 被称为“便笺式”的科学工程计算语言。

MATLAB 的最重要的特征是它拥有解决特定应用问题的程序组，也就是 TOOLBOX（工具箱），如信号处理工具箱、控制系统工具箱、神经网络工具箱、模糊逻辑工具箱、Simulink 工具箱、通信工具箱和数据采集等许多专用工具箱，对大多数用户来说，要想灵活、高效地运用这些工具箱，通常都需要学习相应的专业知识。



此外，开放性也许是 MATLAB 最重要和最受欢迎的特点之一。除内部函数外，所有的 MATLAB 主要文件和各工具箱文件都是可读的、可改的源文件，因为工具箱实际上是由一组复杂的 MATLAB 函数（M 文件）组成，它扩展了 MATLAB 的功能，用以解决特定的问题，因此用户可以通过对源文件进行修改和加入自己编写的文件去构建新的专用工具箱。

1.1.3 MATLAB 系统组成

（1）MATLAB 开发环境

这是一组工具和程序，帮助用户使用 MATLAB 功能和文件。许多工具是图形用户界面，包括 MATLAB 桌面和命令窗口，命令的历史窗口，编辑器和查错程序，观看帮助信息的浏览器，工作区，文件和搜索路径。

（2）MATLAB 语言

这是一种高级的矩阵/数组编程语言，该语言带有流程控制语句、函数、数据结构、输入/输出和面向对象编程的特点。它既可以编写快速执行的短小程序，也可以编写庞大的复杂应用程序。

（3）MATLAB 图形处理系统

这是 MATLAB 的图形系统，它既包括生成二维和三维数据可视化、图像处理、动画及演示图形的高级命令，也包括完全由用户自定制图形显示及在 MATLAB 应用程序中创建完整的图形用户接口的低级命令。

（4）MATLAB 的数学函数库

这是一个计算算法的巨大集合，范围从初等函数，如求和、正弦、余弦和复数运算，到更高级的函数，像矩阵求逆、矩阵特征值、贝塞尔函数和快速傅里叶变换。

（5）MATLAB 应用程序接口（API）

这是一个用户编写的 MATLAB 接口的 C 语言和 FORTRAN 语言程序的函数库，它包括从 MATLAB（动态链接）中调用命令和读写 M 文件的程序。

1.1.4 MATLAB R2009a 的新特点

MATLAB R2009a 于 2009 年 3 月正式发布，TMW 正式发布了 MATLAB R2009a，新版本涵盖：Simulink 8、新产品 Simulink Design Verifier、Link for Analog Devices VisualDSP 以及 82 个产品模块的更新升级及 Bug 修订。从现在开始，MathWorks 公司将每年进行两次产品发布，时间分别在每年的 3 月和 9 月，而且每一次发布都会包含所有的产品模块，如产品的 new feature、bug fixes 和新产品模块的推出。在 R2008b 中（MATLAB 7.6，Simulink 6.4），主要更新了多个产品模块、增加了多达 350 个新特性、增加了对 64 位 Windows 的支持，并新推出了.NET 工具箱。

作为和 Mathematica、Maple 并列的三大数学软件，其强项就是其强大的矩阵计算以及仿真能力。要知道，MATLAB 的由来就是 MATrix + LABoratory = MATLAB，所以这个软件在国内也被称作“矩阵实验室”。每次 MathWorks 发布 MATLAB 的同时也会发布仿真工具 Simulink。在欧美很多大公司在将产品投入实际使用之前都会进行仿真试验，主要使用的仿真软件就是 Simulink。MATLAB 提供了自己的编译器：全面兼容 C++以及 FORTRAN 两大语言。所以 MATLAB 是工程师、科研工作者手上最好的语言，最好的工具和环境。MATLAB 已经成为广大科研人员的最值得信赖的助手。

MATLAB R2009a 完整版，这次的升级做了重大的增强改进，也升级了以下各版本，提供了 MATLAB，Simulink 的升级以及其他最新模块的升级。这个 MATLAB 2008 版本不仅仅提高了产品质量，同时也提供了新的用于数据分析、大规模建模、固定点开发、编码等新特征。

其中，MATLAB Builder for .NET 扩展了 MATLAB Compiler 的功能，主要有：

- 可以打包 MATLAB 函数，使网络程序员可以通过 C#，VB.NET 等语言访问这些函数。
- 创建组件来保持 MATLAB 的灵活性。
- 创建 COM 组件。
- 将源自 MATLAB 函数的错误作为一个标准的管理异常来处理。

MATLAB R2009a 新版本中，产品模块进行了一些调整，MATLAB Builder for COM 的功能集成到 MATLAB Builder for .net 中去，Finacial Time Series Toolbox 的功能集成到 Financial Toolbox 中。MATLAB 将高性能的数值计算和可视化集成在一起，并提供了大量的内置函数，从而被广泛地应用于科学计算、控制系统、信息处理等领域的分析、仿真和设计工作，而且利用 MATLAB 产品的开放式结构，可以非常容易地对 MATLAB 的功能进行扩充，从而在不断深化对问题认识的同时，不断完善 MATLAB 产品以提高产品自身的竞争能力。

1.2 MATLAB R2009a 的基本操作

1.2.1 MATLAB R2009a 的安装与激活

MATLAB R2009a 在安装过程上与 MATLAB R2008 在安装与激活上基本相同，都增加了对 MATLAB 的激活环节。具体安装步骤如下：

1）将 MATLAB R2009a 的安装盘放入 CD-ROM 驱动器，系统将自动运行程序，进入初始化界面，如图 1-1 所示。

2）启动安装程序后显示的安装界面如图 1-2 所示。选中“Install manually without using the Internet”单选按钮，再单击“Next”按钮。

3）系统弹出如图 1-3 所示的“License Agreement”（查看软件注册协议）窗口，若同意 Math Works 公司的安装许可协议，单击“Yes”单选按钮，再单击“Next”按钮。

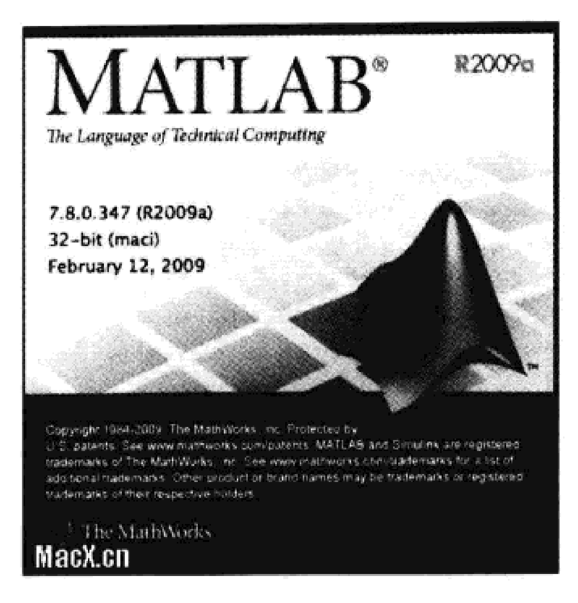


图 1-1　MATLAB R2009a 安装的启动界面

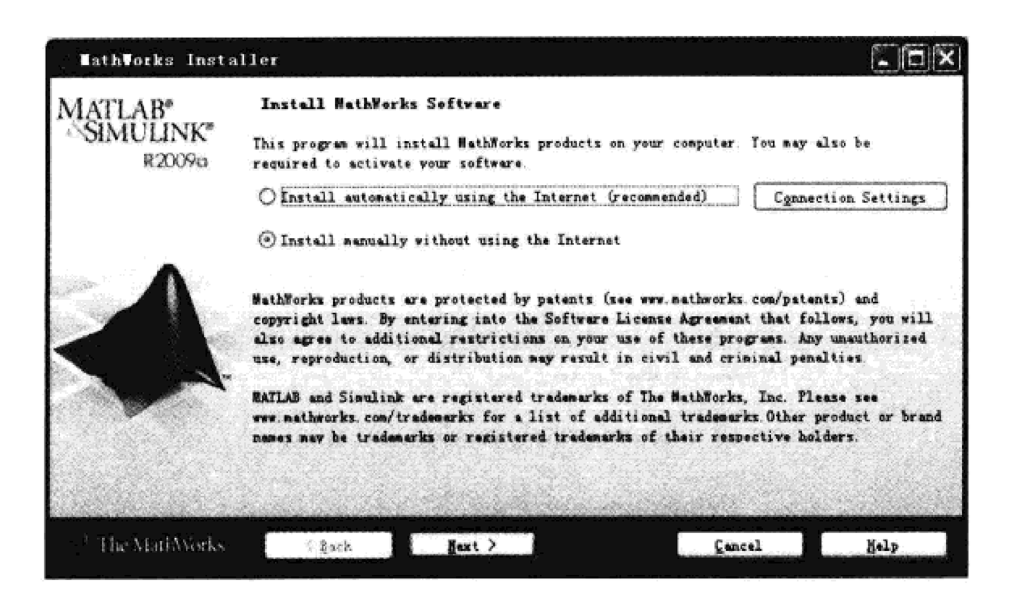


图 1-2　“MathWorks Installer”安装界面

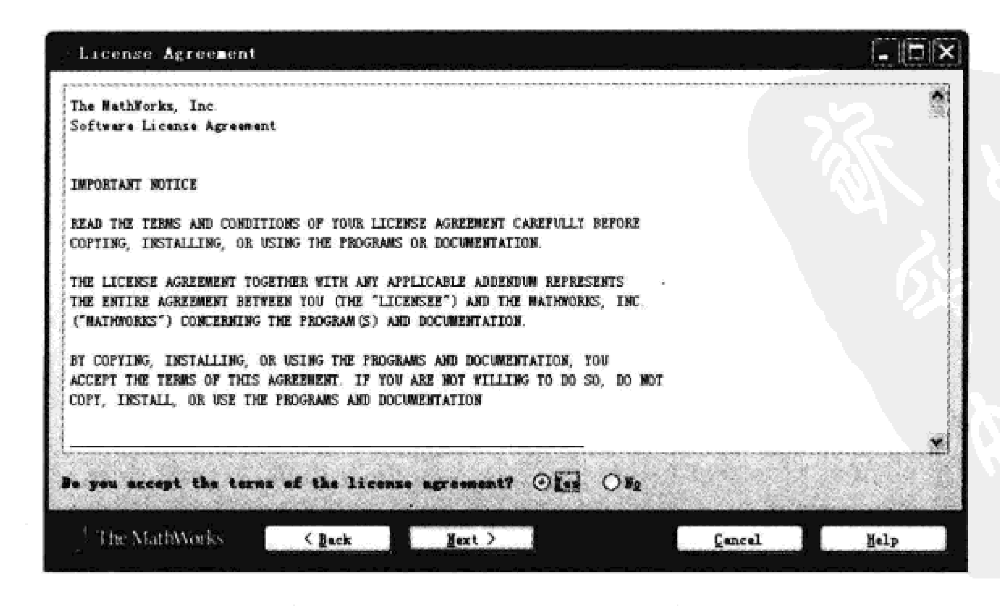


图 1-3　“License Agreement”窗口



4）系统弹出如图 1-4 所示的“File Installation Key”窗口，输入软件外包装封面或安装许可文件内提供的密钥，单击“Next”按钮。

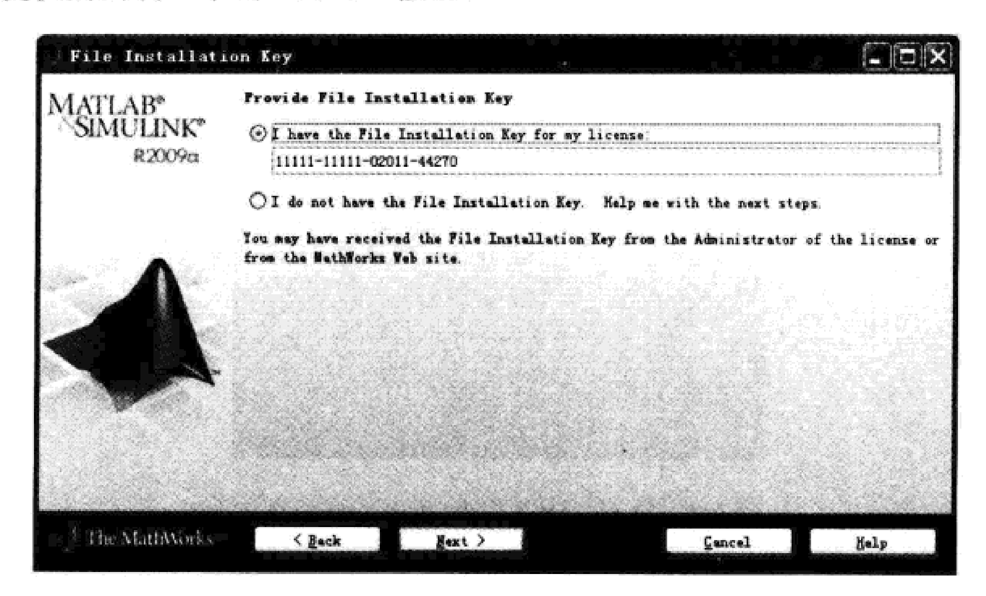


图 1-4 “File Installation Key”窗口

5）若输出正确的密钥，系统将弹出如图 1-5 所示的“Installation Type”窗口，可以选择“Typical”或“Custom”安装类型。如果选择“Typical”单选按钮，MATLAB R2009a 安装工具默认安装所有工具箱及组件，此时所需磁盘空间超过 6GB。

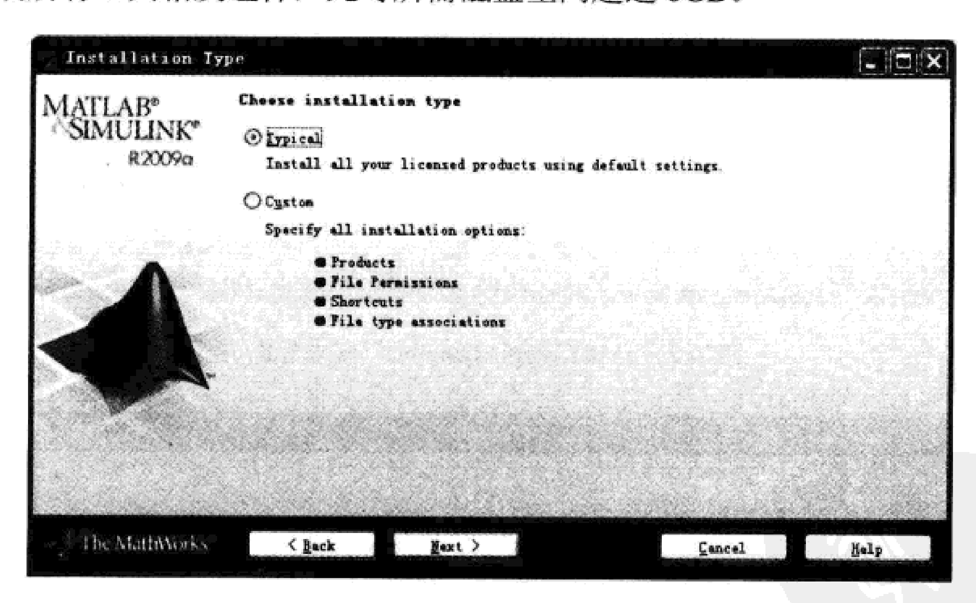


图 1-5 “Installation Type”窗口

6）默认路径为 C:\Program File\MATLAB\R2009。用户可以通过单击“Browser”按钮选择其他安装文件夹，如作者选择安装在“F:\MATLAB R2009”目录下，如果 F 盘下没有“MATLAB R2009”文件夹，安装程序自动建立，此时“Folder Selection”窗口的下部将显示安装硬盘剩余空间及软件安装所需空间大小（图示为全部安装所需软件大小）。单击“Next”按钮，如图 1-6 所示。

7）确定安装路径的下一步，系统将弹出如图 1-7 所示的“Confirmation”窗口，可以看到用户所默认安装的 MATLAB 组件、安装文件夹等相关信息。单击“Install”按钮，开始安装。



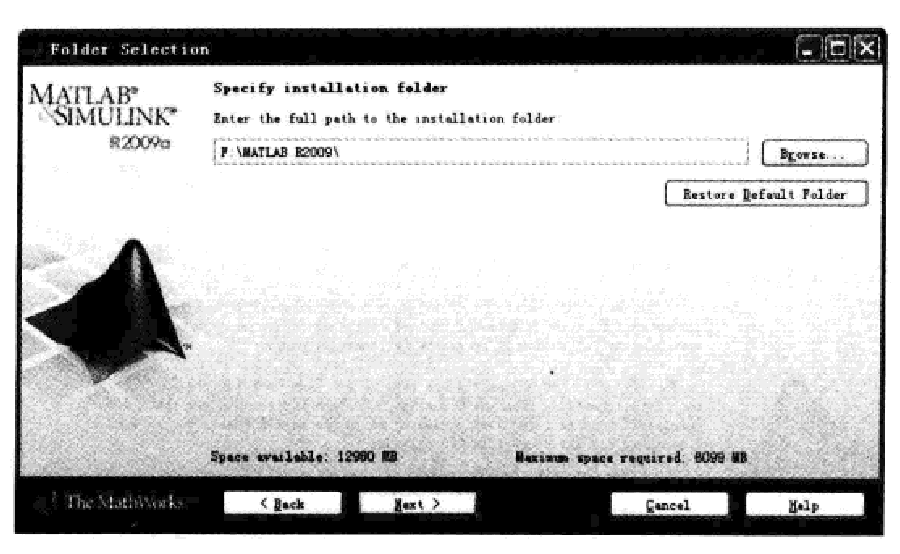


图 1-6 “Folder Selection”窗口

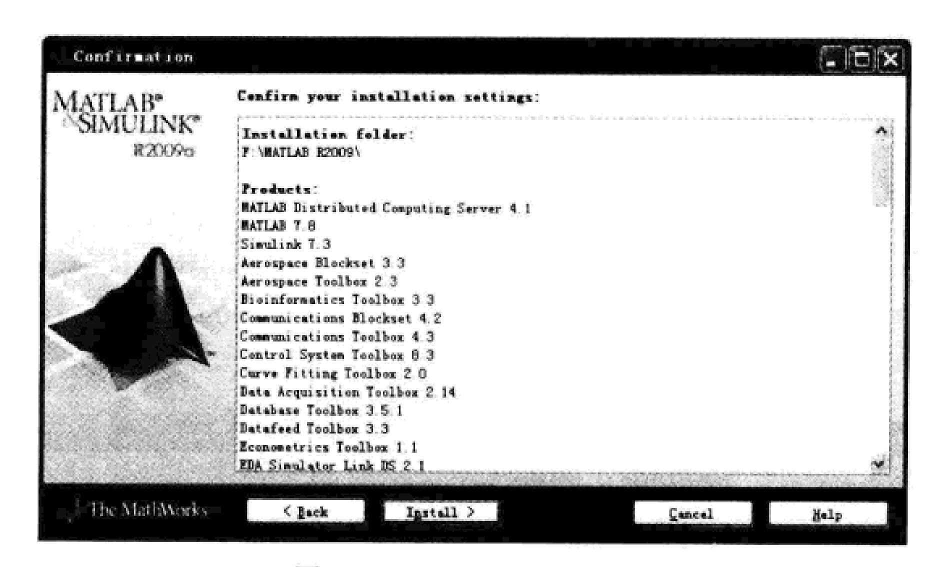


图 1-7 “Confirmation”窗口

8）软件在安装过程中，将显示安装进度条如图 1-8 所示。用户需要等待产品组件安装完成，同时可以查看正在安装的产品组件及安装剩余的时间。安装完成后，系统弹出如图 1-9 所示的“Product Configuration Notes”窗口。

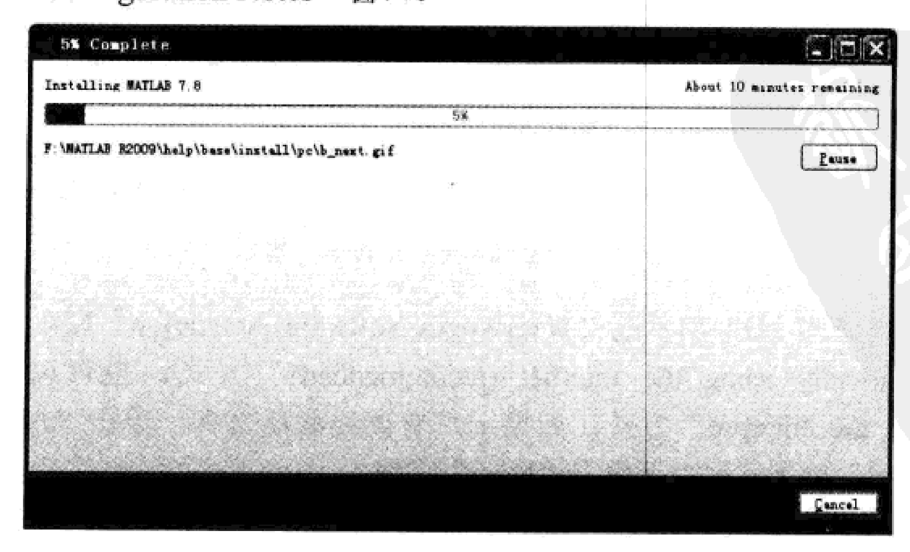
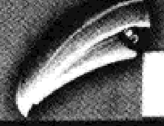


图 1-8 安装进度窗口



9）在安装完产品组件之后，MathWorks 公司需要用户进行产品配置。在如图 1-9 所示的“Product Configuration Notes”窗口中，单击“Next”按钮。

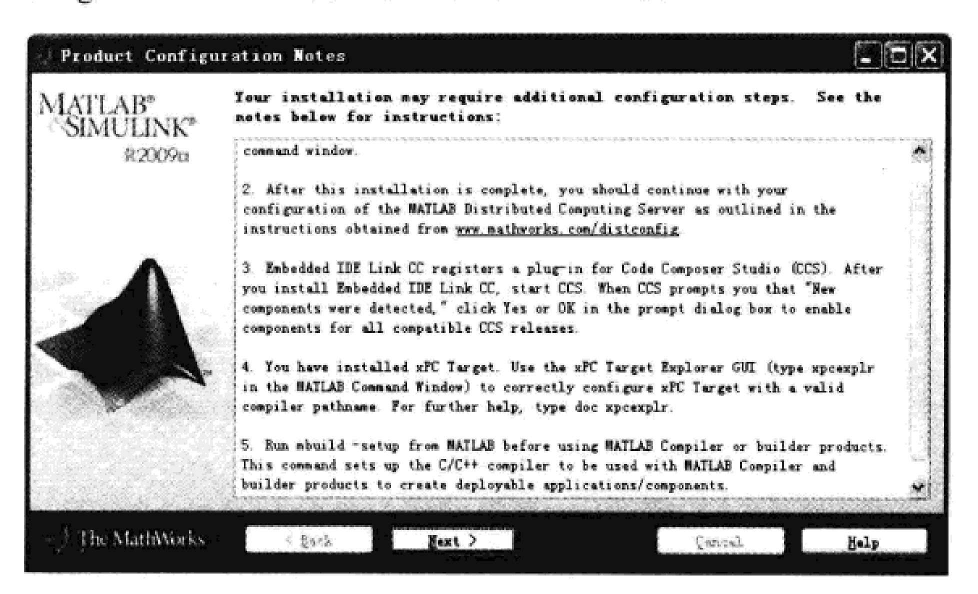


图 1-9 “Product Configuration Notes”窗口

10）安装结束后，系统将显示一个如图 1-10 所示“Installation Complete”窗口，用户需要进行 MATLAB 软件的激活操作，否则软件不能使用，这是 MathWorks 公司为了保护知识产权从 MATLAB R2008a 起新增设的保护措施。MATLAB R2009 也具有这种保护措施。此时 MATLAB 软件的安装已经完成，单击“Next”按钮，进行软件激活。

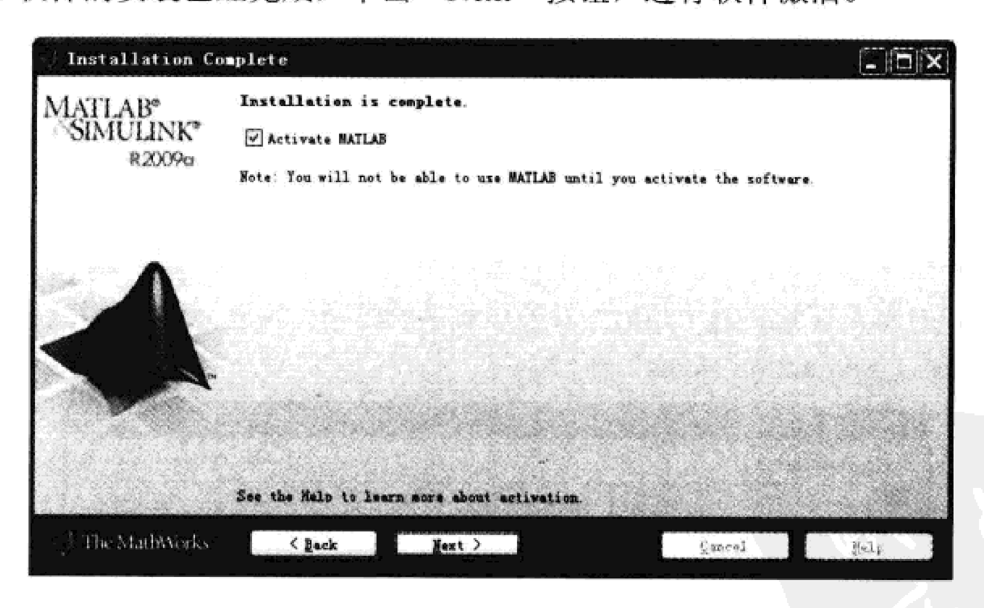


图 1-10 “Installation Complete”窗口

11）系统弹出如图 1-11 所示的“MathWorks Software Activation”窗口，用户可以选择“Activate automatically using the Internet (recommended)”方式，也可以选择“Activate manually without the Internet”方式。如果用户有离线激活文件，单击“Activate manually without the Internet”单选按钮，再单击“Next”按钮。

12）系统弹出如图 1-12 所示的“Offline Activation”对话框，用户选择离线激活许可文件，单击“Next”按钮，系统弹出如图 1-13 所示的“Activation Complete”窗口。

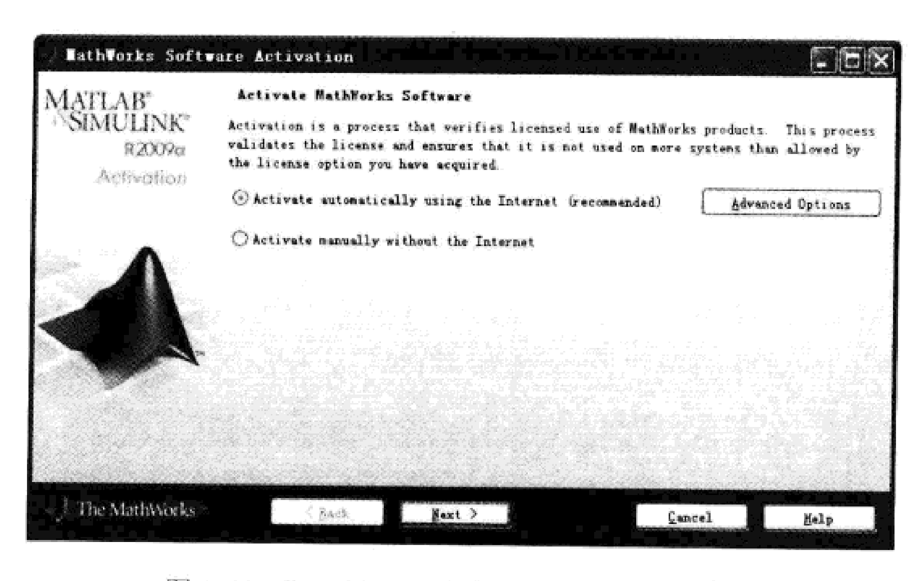


图 1-11 “MathWorks Software Activation”窗口

13）单击图 1-12 中的“Activation Complete”窗口中的“Finish”按钮即可。

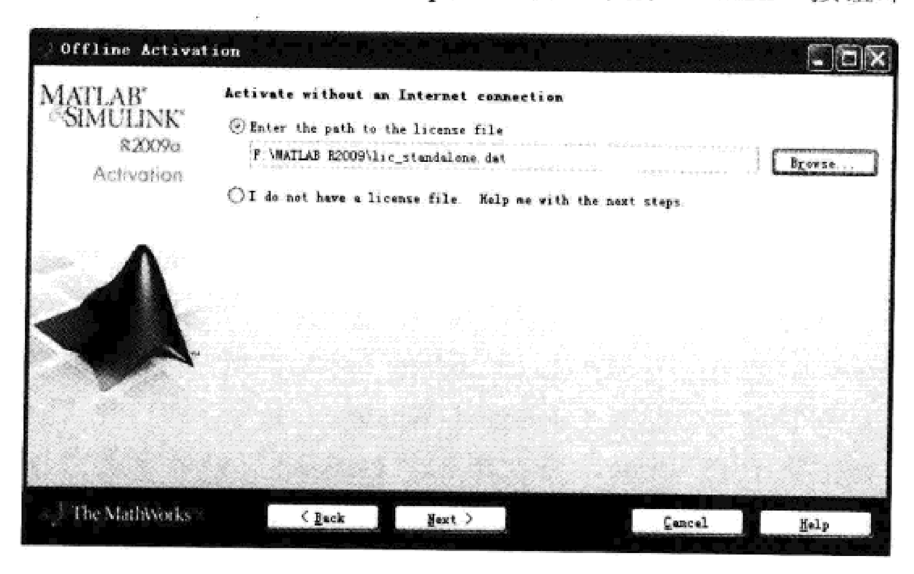


图 1-12 “Offline Activation”窗口

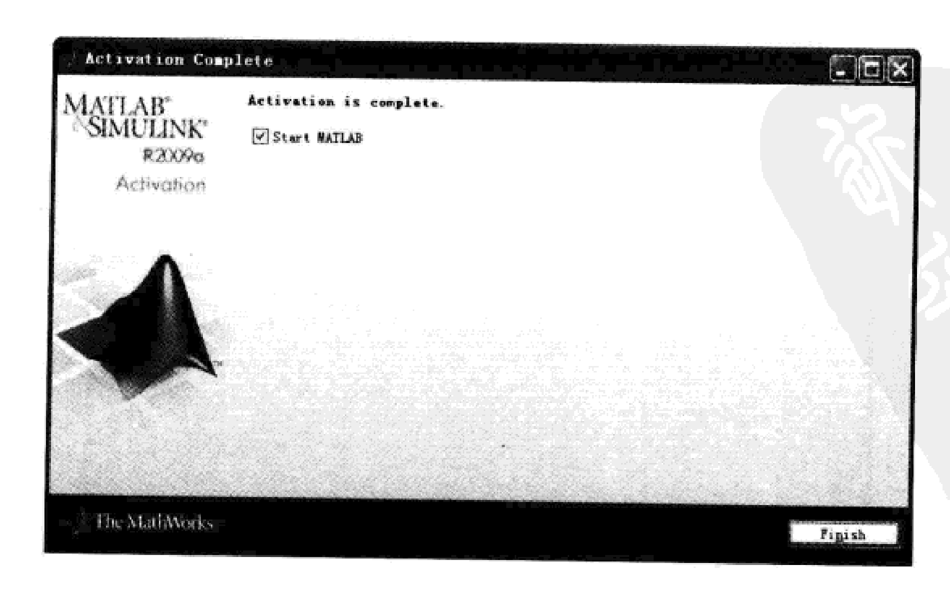


图 1-13 “Activation Complete”窗口



1.2.2 操作界面

用鼠标双击桌面上的 MATLAB R2009a 快捷方式图标，可打开如图 1-14 所示的基本操作界面。其操作界面与大家熟悉的 Windows 视窗操作系统非常接近，这有助于用户适应该软件。除了菜单栏、工具栏和一个产品更新信息提示栏之外，MATLAB R2009a 基本界面包括 Command Window（命令窗口）、Workspace（工作空间游览器）、Command Histroy（历史命令浏览器）和 Current Directory（搜索路径与当前目录）。

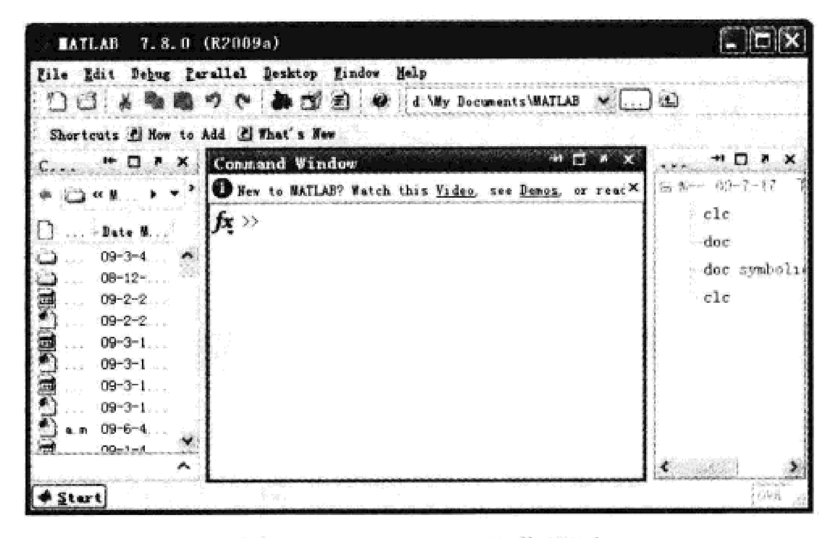


图 1-14 MATLAB 工作界面

1. Command Window

Command Window 是用户与 MATLAB 交互的主窗口，用户可以将其放大。单击菜单栏中的“Desktop”菜单下的“Undock Command Window”命令，即可独立地打开“Command Window”窗口，如图 1-15 所示。再次单击“Desktop”菜单下的“Undock Command Window”命令，Command Window 恢复到刚打开的默认状态。MATLAB 将当前窗口的标题栏置为深蓝色，其他非当前窗口为灰色。如果初次使用 MATLAB R2009a，在 Command Window 菜单栏下有一行提示“New to MATLAB?”，如果用户初始使用该软件，可以单击“Watch this Video”、“See Demos”或“Read Getting Started”去学习 MATLAB 软件的基本操作方法。如果用户对 MATLAB 软件的基本操作方法足够熟悉，可以单击“☒”按钮将其关闭。

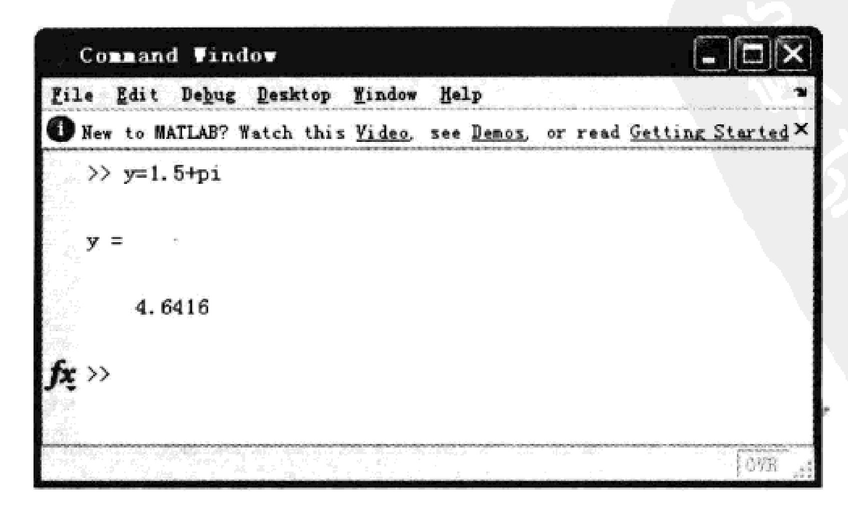


图 1-15 “Command Window”窗口

在“Command Window”窗口中，“fx >>”是命令提示符（是 MATLAB R2009a 特有的），在它之后可以直接输入命令，如图 1-15 所示。MATLAB 将命令执行结果显示在窗口中，便于用户查看。MATLAB 对窗口中的命令逐行解释执行，如果有多条命令，可以逐行输入，也可以在同一行里输入多条命令，它们之间用逗号隔开或采用将在下面介绍的 M 文件输入。当一行命令太长无法在窗口一次输入完时，可以使用“...”将命令续行。



常用的窗口操作命令有 clear（清除工作空间变量）、clc（清除命令窗口的内容但不清除工作空间变量）和 clf（清除当前图形窗口的内容），键盘方向键〈↑〉和〈↓〉可以用来搜寻 Command Window 执行过的命令。熟练使用窗口操作的基本命令，有助于快编程速度，其他窗口操作命令在下面章节将陆续介绍到。

2. Workspace

Workspace 也称为内存空间浏览器，它保存了命令窗口所使用过的全部变量，可通过 Workspace 对内存变量进行操作。单击菜单栏上“Desktop”菜单下的“Undock Command Window”命令，可弹出独立的“Workspace”窗口，如图 1-16 所示。单击“Workspace”窗口中的某个内存变量，可对其进行复制、删除等操作。内存空间的变量在使用 clear 命令后将被清除。

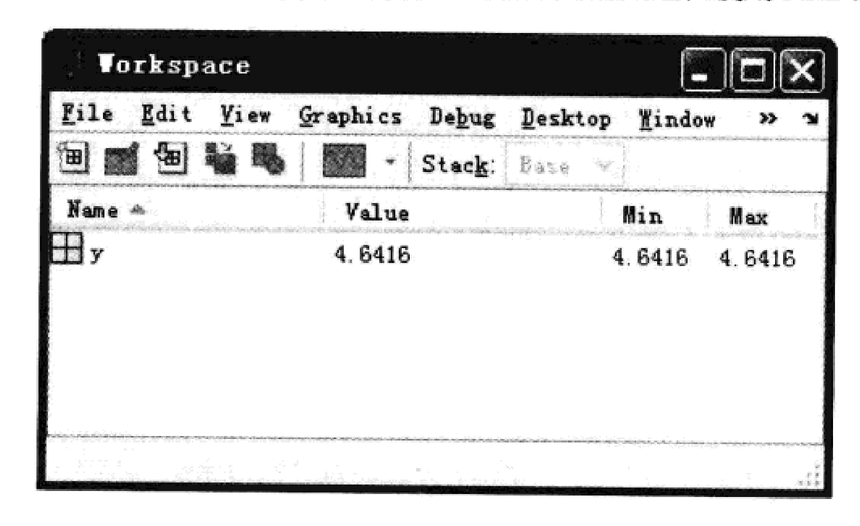


图 1-16 “Workspace”窗口

双击内存空间中的变量，还可以打开相应的矩阵编辑器，可直接对变量进行编辑操作，如图 1-17 所示。在这里可以方便地输入大矩阵或对矩阵局部元素进行修改。

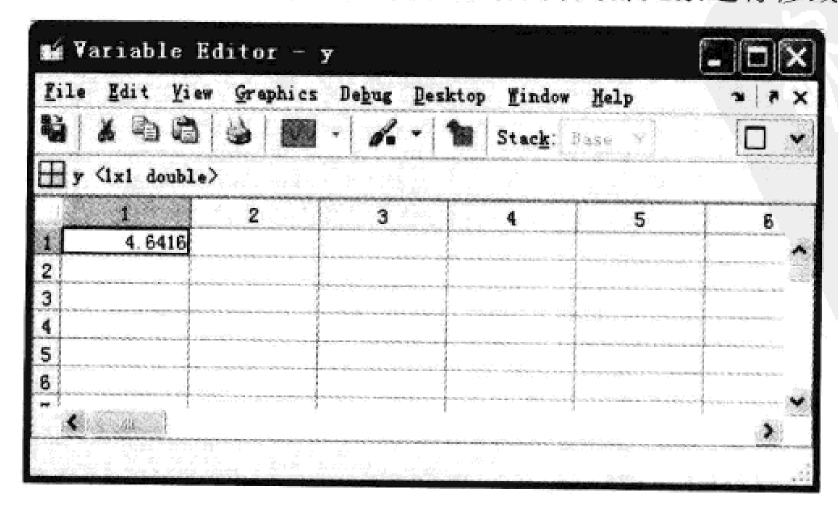


图 1-17 变量编辑器

3. Command Histroy

Command Histroy 记录着用户在 Command Window 输入过的所有命令，选中该命令后单击鼠标右键，在弹出的快捷菜单中，可对命令进行剪切、复制、删除及 Command Window 执行等操作。单击菜单栏上的"Desktop"菜单下的"Undock Command Window"命令，也可以将其单独弹出，如图 1-18 所示。

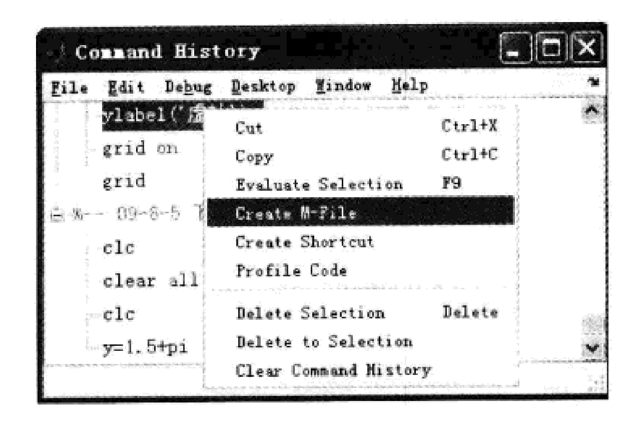


图 1-18 "Command Histroy"窗口

4. Current Directory

MATLAB 的所有文件都放在一组目录（文件夹）上。MATLAB 把这些目录按优先级设计为"搜索路径"上的结点，此后 MATLAB 工作时，就沿着此搜索路径，从各个目录上寻找所需的文件、函数和数据。

当 Command Window 输入一条命令后，MATLAB 对该命令的基本搜索过程是："是否为为内存变量"→"是否为内建函数"→"是否为当前目录上的 M 文件"→"是否为 MATLAB 路径上其他目录的 M 文件"。如果在搜索路径上存在同名函数，则 MATLAB 仅发现搜索路径中的第一个函数，而其他同名函数不被执行。

使用以下命令可对当前的搜索路径进行操作。

- Path：不加任何参数，显示当前搜索路径。
- Path[路径名]：设置当前搜索路径，以前的搜索路径无效，如 Path D:\MATLAB R2009a\bin。
- addpath D:Mywork：向当前搜索路径中添加目录 D:Mywork。
- rmpath D:Mywork：取消当前搜索路径中的目录 D:Mywork。

此外，使用 Pathtool 命令或单击"File"菜单下的"Set Path"命令，可打开如图 1-19 所示的"Set Path"窗口，进行搜索路径的设置。

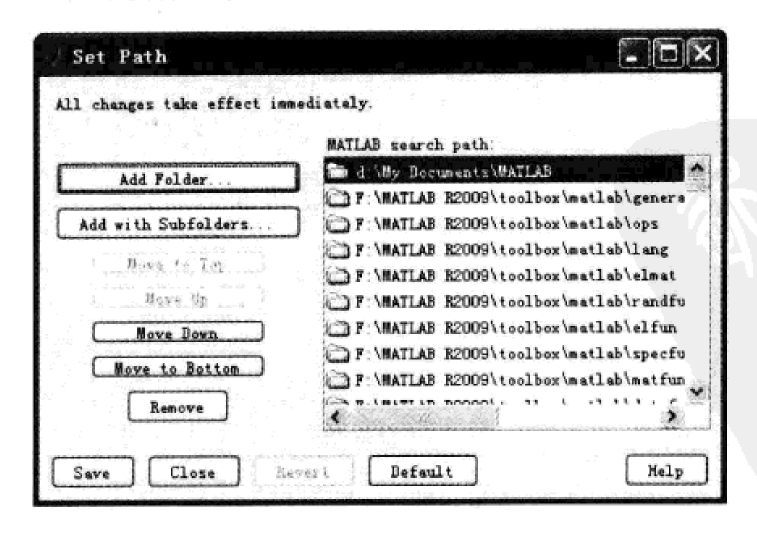


图 1-19 "Set Path"窗口

MATLAB 文件的打开与保存等操作，默认地址位于 MATLAB 默认的当前目录路径下。MATLAB 默认的当前目录路径为"D:\My Documents\MATLAB"（D 为 MATLAB 的安装盘

符）。默认的当前目录窗口如图 1-20 所示。

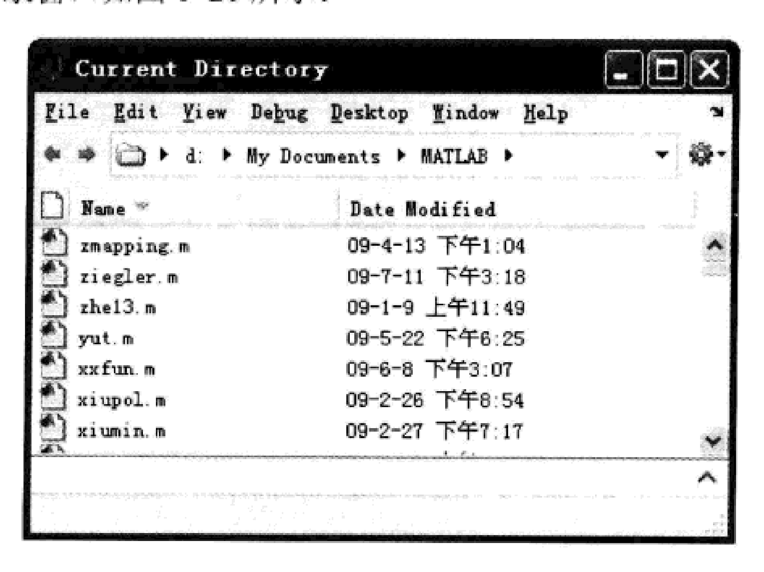


图 1-20　当前目录窗口



用户可以将自己的目录永久地设置在 MATLAB 的搜索路径上，也可以将自己的目录设置为 MATLAB 的当前目录。用户目录设置为当前目录的方法如下：

1）在目录浏览器的“”列表框中直接填写待设置的目录名，或借助目录浏览器右边的浏览按钮选择待设定的目录。

2）在 Command Window 中用命令设置，如在 Command Window 中输入：

```
>> mkdir d:\MyMATLABfile   %在 D 盘下创建名为 MyMATLABfile 的文件夹
>> cd d:\MyMATLABfile   %将 D 盘下名为 MyMATLABfile 的文件夹设置为当前目录
```

1.2.3 帮助系统

1．帮助命令

要了解 MATLAB，最简洁快速的方式是在命令窗口通过帮助命令对特定的内容进行快速查询。帮助命令包括 help 命令和 lookfor 命令。

（1）help 命令

help 命令是查询函数语法的最基本方法，查询信息直接显示在命令窗口中。在命令窗口中直接输入 help 命令将会显示当前帮助系统中所包含的所有项目，即搜索路径中所有的目录名称。

同样，可以通过 help 加函数名来显示该函数的帮助说明。例如，为了显示 magic 函数的使用方法与功能，可以使用以下命令。

```
>> help magic
```

屏幕显示帮助信息：

```
>> help solve
 SOLVE   Symbolic solution of algebraic equations.
    SOLVE('eqn1','eqn2',...,'eqnN')
    SOLVE('eqn1','eqn2',...,'eqnN','var1,var2,...,varN')
```

```
SOLVE('eqn1','eqn2',...,'eqnN','var1','var2',...'varN')
  The eqns are symbolic expressions or strings specifying equations.  The
vars are symbolic variables or strings specifying the unknown variables.
SOLVE seeks zeros of the expressions or solutions of the equations.
If not specified, the unknowns in the system are determined by FINDSYM.
If no analytical solution is found and the number of equations equals
the number of dependent variables, a numeric solution is attempted.
Three different types of output are possible.  For one equation and one
output, the resulting solution is returned, with multiple solutions to
a nonlinear equation in a symbolic vector.  For several equations and
an equal number of outputs, the results are sorted in lexicographic
order and assigned to the outputs.  For several equations and a single
output, a structure containing the solutions is returned.
Examples:
    solve('p*sin(x) = r') chooses 'x' as the unknown and returns
      ans =
      asin(r/p)
    [x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0') returns
      x =[ 1]
        [ 3]
      y = [       1]
         [ -3/2]
     S = solve('x^2*y^2 - 2*x - 1 = 0','x^2 - y^2 - 1 = 0') returns
    the solutions in a structure.
       S =
         x: [8x1 sym]
         y: [8x1 sym]
     [u,v] = solve('a*u^2 + v^2 = 0','u - v = 1') regards 'a' as a
    parameter and solves the two equations for u and v.
     S = solve('a*u^2 + v^2','u - v = 1','a,u') regards 'v' as a
    parameter, solves the two equations, and returns S.a and S.u.
     [a,u,v] = solve('a*u^2 + v^2','u - v = 1','a^2 - 5*a + 6') solves
    the three equations for a, u and v.
  See also dsolve.
Overloaded methods:
    cgvariable/solve
    cgvalue/solve
    cgsubexpr/solve
    cgfeature/solve
    cgexpr/solve
    cgdivexpr/solve
Reference page in Help browser
    doc solve
```

值得注意的是，MATLAB 命令窗口里显示的帮助信息用大写来突出函数名，但在使用函数时，要用小写。

MATLAB 按照函数的不同用途分别存放在不同的子目录下，用相应的帮助命令可显示某一类函数。例如，所有的线性代数函数均收在 matfun 子目录下，用命令：

```
>> help matfun
```



可显示所有线性代数函数。

（2）lookfor 命令

help 命令只搜索出那些关键字完全匹配的结果，lookfor 命令对搜索范围内的 M 文件进行关键字搜索，条件比较宽松。例如，因为不存在 inverse 函数，命令：

```
help inverse
```

搜索结果为：

```
inverse not found.
```

而执行命令：

```
>> lookfor inverse
```

将得到 M 文件中包含 inverse 的全部函数。

lookfor 命令只对 M 文件的第一行进行关键字搜索。若在 lookfor 命令加上“all”选项，则可对 M 文件进行全文搜索。

2. 帮助窗口

MATLAB 为用户提供了详细的帮助系统，如 MATLAB 的在线帮助、PDF 格式的帮助文件及 MATLAB 的演示系统等。获取帮助的方法很多，单击 MATLAB 主窗口上的“Help”按钮或单击“Help”菜单下的“Product Help”命令或在命令窗口中输入 help 命令或按〈F1〉快捷键都能打开帮助系统，如图 1-21 所示。MATLAB 帮助系统是学习掌握 MATLAB 软件的最佳工具，它提供了大量帮助说明信息、仿真模型实例和演示系统，唯一的缺点是当前没有中文版的帮助系统，需要用户具备一定的英语阅读能力。

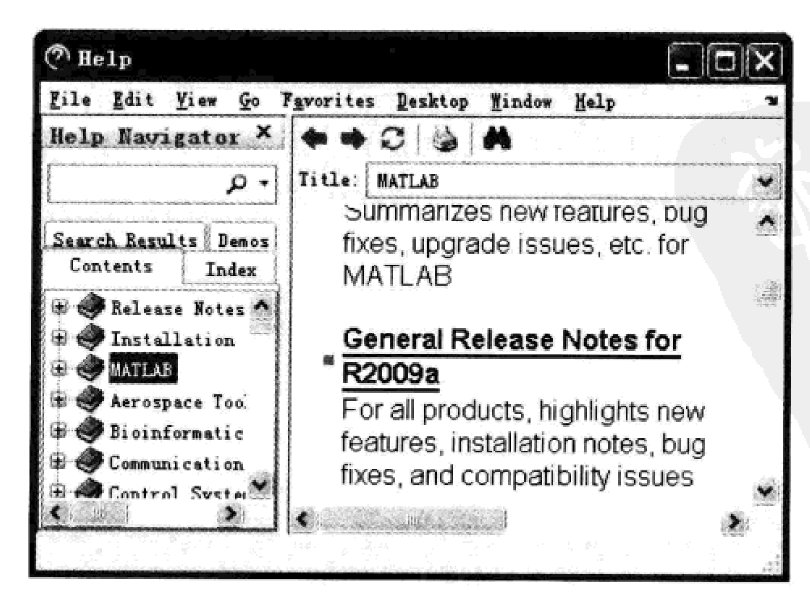


图 1-21 MATLAB 帮助系统

打开帮助系统后，如果要查找某个函数的帮助信息，可通过 Contents 查看相应的主题，也可以在主题栏中输入函数名进行搜索或通过学习 Index 信息进行查询。MATLAB 还提供了丰富的演示系统，有助于用户学习了解 MATLAB 建模与仿真的基本方法。例如，要查询有关控制系统的时域响应函数，首先单击“Demos”选项卡，在弹出的页面中依次单击树形目录“Toolboxes”→“Control System”→“Model Analysis”前的“+”节点，系统显示如图 1-22 所示的窗口。在窗口的右侧，列出了求取控制系统时域响应的基本内容，再单击具有下画线的各关键词，可打开与其相链接的内容。

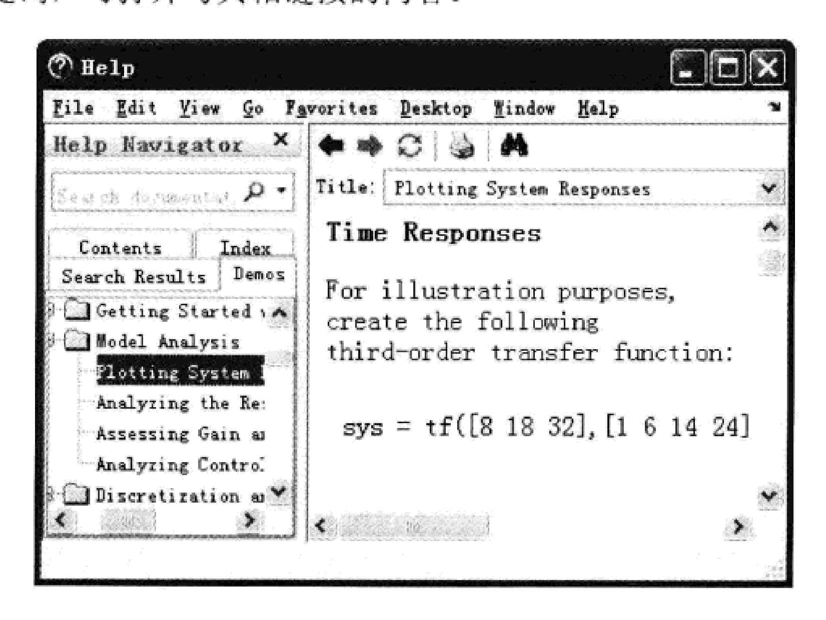


图 1-22　MATLAB 的帮助演示系统

Math Works 公司每年 3 月和 9 月都将发布新的版本。对于版本更新、技术支持等问题，用户可以直接访问有关 MATLAB 的站点进行查询，如 www.mathworks.com。

1.3　MATLAB 的语言特点

1.3.1　MATLAB 语言结构

MATLAB 命令窗口就是 MATLAB 语言的工作空间，因为 MATLAB 的各种功能的执行必须在此窗口下才能实现。在这种环境下输入的 MATLAB 语句称作“窗口命令”。所谓窗口命令就是在上述环境下输入的 MATLAB 语句，直接执行它们完成相应的运算及图等。

MATLAB 语句的一般格式为：

变量=表达式;

式中，等号右边的表达式可由操作符或其他字符、函数和变量组成，它可以是 MATLAB 允许的数学或矩阵运算，也可以包含 MATLAB 下的函数调用；等号左边的变量名为 MATLAB 语句右边表达式的返回值语句所赋值的变量的名字。在调用函数时 MATLAB 允许一次返回多个结果，这时等号左边的变量名需用“[ ]”括起来，且各个变量名之间用逗号分隔开。如果左边的变量名默认时，则返回值自动赋给变量 ans。

在 MATLAB 中变量名必须以字母开头，之后可以是任意字母、数字或者下画线（不能超过 64 个字符），但变量中不能含有标点符号。变量名区分字母的大小写，同一名字的大写与小写被视为两个不同的变量。一般说来，在 MATLAB 下变量名可以为任意字符串，但 MATLAB 保留了一些特殊的字符串常量，见表 1-1。



表 1-1　常用的数学常量

常量符号	常量含义	常量符号	常量含义
eps	浮点数相对精度	pi	圆周率
i	虚数实部单位	inf	非数值
j	虚数虚部单位	NaN	正无穷
realmin	最小正浮点数	realmax	最大正浮点数

MATLAB 是一种类似 Basic 语言的解释性语言，命令语句逐条解释逐条执行，它不是输入全部 MATLAB 命令语句，并经过编译、连接形成可执行文件后才开始执行的，而是每输入完一条命令，在输入〈Enter〉键后 MATLAB 就立即对其处理，并得出中间结果，完成了 MATLAB 所有命令语句的输入，也就完成了它的执行，直接得到最终结果。从这一点来说，MATLAB 清晰地体现了类似“演算纸”的功能。

例如：

```
>> A=2;
>> B=7;
>> C=A*B
```

运行程序，输出结果如下：

```
C =
     14
```

运行程序，输出结果如下：

```
>> C2=C+3
C2 =
     17
```

MATLAB 语句既可由分号结束，也可由逗号或换行结束，但它的含义不同。用分号“；”结束（半角状态的分号），则说明执行了这一条命令，MATLAB 这时将不立即显示运行的中间结果，而是等待下一条命令的输入，以上前两条命令如果以逗号“，”或“回车”结束，则将把左边变量的值全部显示在屏幕上。当然在任何时候也可输入相应的变量名来查看其内容。

例如：

```
>> A=2
```

运行程序，输出结果如下：

```
A =
     2
```

在 MATLAB 中，几条语句也可以出现在同一行中，只要用分号或逗号将它们分割即可。

例如：

```
>> A=2;B=7;C=A*B;C2=C+3
```

运行程序，输出结果如下：

```
C2 =
     17
```

这个命令得到以上相同的结果。

1.3.2 MATLAB 常用命令操作

在 MATLAB 工作空间中，通过 MATLAB 常用命令实现对空间的管理、在线帮助等功能。

1. MATLAB 的内容

MATLAB R2009a 是功能特别强大、特别全的大型系统软件，内容极为丰富。首先在安装 MATLAB R2009a 的过程中，就会见到很多的组件，令人眼花缭乱；其次，执行命令“ver”也能列出系统组件名、组件版本号。这些组件分别隶属于 MATLAB R2009a 的子目录里。请看以下示例。

【例 1-1】 运行以下 DOS 命令显示 MATLAB R2009a 的系统组件名、组件版本号。

```
>> ver
-------------------------------------------------------------------------------------
MATLAB Version 7.8.0.347 (R2009a)
MATLAB License Number: 161051
Operating System: Microsoft Windows XP Version 5.1 (Build 2600: Service Pack 2)
Java VM Version: Java 1.6.0_04-b12 with Sun Microsystems Inc. Java HotSpot(TM) Client VM mixed
mode
-------------------------------------------------------------------------------------
MATLAB                                   Version 7.8           (R2009a)
Simulink                                 Version 7.3           (R2009a)
Aerospace Blockset                       Version 3.3           (R2009a)
Aerospace Toolbox                        Version 2.3           (R2009a)
Bioinformatics Toolbox                   Version 3.3           (R2009a)
Communications Blockset                  Version 4.2           (R2009a)
Communications Toolbox                   Version 4.3           (R2009a)
Control System Toolbox                   Version 8.3           (R2009a)
Curve Fitting Toolbox                    Version 2.0           (R2009a)
Data Acquisition Toolbox                 Version 2.14          (R2009a)
Database Toolbox                         Version 3.5.1         (R2009a)
Datafeed Toolbox                         Version 3.3           (R2009a)
EDA Simulator Link DS                    Version 2.1           (R2009a)
EDA Simulator Link IN                    Version 2.4           (R2009a)
```

EDA Simulator Link MQ	Version 2.6	(R2009a)
Econometrics Toolbox	Version 1.1	(R2009a)
Embedded IDE Link CC	Version 3.4	(R2009a)
Embedded IDE Link MU	Version 1.2	(R2009a)
Embedded IDE Link TS	Version 1.4	(R2009a)
Embedded IDE Link VS	Version 2.2	(R2009a)
Filter Design HDL Coder	Version 2.4	(R2009a)
Filter Design Toolbox	Version 4.5	(R2009a)
Financial Derivatives Toolbox	Version 5.4	(R2009a)
Financial Toolbox	Version 3.6	(R2009a)
Fixed-Income Toolbox	Version 1.7	(R2009a)
Fixed-Point Toolbox	Version 2.4	(R2009a)
Fuzzy Logic Toolbox	Version 2.2.9	(R2009a)
Gauges Blockset	Version 2.0.5	(R2009a)
Genetic Algorithm and Direct Search Toolbox	Version 2.4.1	(R2009a)
Image Acquisition Toolbox	Version 3.3	(R2009a)
Image Processing Toolbox	Version 6.3	(R2009a)
Instrument Control Toolbox	Version 2.8	(R2009a)
MATLAB Builder EX	Version 1.2.12	(R2009a)
MATLAB Builder JA	Version 2.0.3	(R2009a)
MATLAB Builder NE	Version 3.0.1	(R2009a)
MATLAB Compiler	Version 4.10	(R2009a)
MATLAB Distributed Computing Server	Version 4.1	(R2009a)
MATLAB Report Generator	Version 3.6	(R2009a)
Mapping Toolbox	Version 2.7.2	(R2009a)
Model Predictive Control Toolbox	Version 3.1	(R2009a)
Model-Based Calibration Toolbox	Version 3.6	(R2009a)
Neural Network Toolbox	Version 6.0.2	(R2009a)
OPC Toolbox	Version 2.1.3	(R2009a)
Optimization Toolbox	Version 4.2	(R2009a)
Parallel Computing Toolbox	Version 4.1	(R2009a)
Partial Differential Equation Toolbox	Version 1.0.14	(R2009a)
RF Blockset	Version 2.4	(R2009a)
RF Toolbox	Version 2.5	(R2009a)
Real-Time Windows Target	Version 3.3	(R2009a)
Real-Time Workshop	Version 7.3	(R2009a)
Real-Time Workshop Embedded Coder	Version 5.3	(R2009a)
Robust Control Toolbox	Version 3.3.3	(R2009a)
Signal Processing Blockset	Version 6.9	(R2009a)
Signal Processing Toolbox	Version 6.11	(R2009a)
SimBiology	Version 3.0	(R2009a)
SimDriveline	Version 1.5.2	(R2009a)
SimElectronics	Version 1.2	(R2009a)
SimEvents	Version 2.4	(R2009a)
SimHydraulics	Version 1.5	(R2009a)
SimMechanics	Version 3.1	(R2009a)



```
SimPowerSystems                         Version 5.1       (R2009a)
Simscape                                Version 3.1       (R2009a)
Simulink 3D Animation                   Version 5.0       (R2009a)
Simulink Control Design                 Version 2.5       (R2009a)
Simulink Design Optimization            Version 1.0       (R2009a)
Simulink Design Verifier                Version 1.4       (R2009a)
Simulink Fixed Point                    Version 6.1       (R2009a)
Simulink HDL Coder                      Version 1.5       (R2009a)
Simulink Report Generator               Version 3.6       (R2009a)
Simulink Verification and Validation    Version 2.5       (R2009a)
Spline Toolbox                          Version 3.3.6     (R2009a)
Spreadsheet Link EX                     Version 3.0.3     (R2009a)
Stateflow                               Version 7.3       (R2009a)
Stateflow Coder                         Version 7.3       (R2009a)
Statistics Toolbox                      Version 7.1       (R2009a)
Symbolic Math Toolbox                   Version 5.2       (R2009a)
System Identification Toolbox           Version 7.3       (R2009a)
SystemTest                              Version 2.3       (R2009a)
Target Support Package FM5              Version 2.2.3     (R2009a)
Target Support Package IC1              Version 1.5.3     (R2009a)
Target Support Package TC2              Version 3.2       (R2009a)
Target Support Package TC6              Version 3.6       (R2009a)
Vehicle Network Toolbox                 Version 1.0       (R2009a)
Video and Image Processing Blockset     Version 2.7       (R2009a)
Wavelet Toolbox                         Version 4.4       (R2009a)
xPC Target                              Version 4.1       (R2009a)
xPC Target Embedded Option              Version 4.1       (R2009a)
```

【例 1-2】 运行以下 DOS 命令显示 MATLAB R2009a 的子目录。

```
>> dir
.            help        lic.txt              patents.txt     toolbox
..           ja          lic_standalone.dat   rtw             trademarks.txt
MATLAB R2009a.lnk   java        license.txt   simulink        uninstall
bin          jhelp       licenses             stateflow
extern       lib         notebook             sys
```

2. 空间管理命令

（1）who 命令

为了查看工作空间中都存在哪些变量名，则可以使用 who 命令来完成。例如，当 MATLAB 的工作空间中有 A、B、C、C2、y 这 5 个变量名时，使用 who 命令操作如下：

```
>> who
```

运行 who 命令，输出结果如下：

```
Your variables are:
A    B    C    C2   y
```

（2）whos 命令

使用 who 命令只能查看到在命令空间的变量列表，可以使用 whos 命令进一步得到变量的详细信息。



```
>> whos
```

运行 whos 命令，输出结果如下：

```
Name      Size         Bytes  Class     Attributes
A         1x1              8  double
B         1x1              8  double
C         1x1              8  double
C2        1x1              8  double
y         1x1              8  double
```

其功能类似在工作空间窗口中显示的变量信息（注意，who 命令与 whos 命令显示的区别）。

（3）clear 命令

了解了当前工作空间中的现有变量名之后，可以使用 clear 命令来删除其中一些不再使用的变量名，这样可能使得整个工作空间更简洁，同时节省一部分内存空间。例如，想删除工作空间中的两个变量，则可以使用下面的命令。

```
>> clear a y
```

然后再查询空间存在的变量，结果如下：

```
>> who
Your variables are:
A    B    C    C2
```

如果想删除整个工作空间中所有的变量，则可以使用以下命令。

```
>> clear
```

这时，再用 who 命令进行查询，注意结果的变化：

```
>> who
>>
```

由此可以看出，如果使用 clear+变量名，即直接删除指定的变量，如果直接使用 clear 命令，即删除工作空间中所有的变量。

（4）clc 命令

在绘制某个程序时，为了保持显示界面的整洁，程序开始第一步应先进行清除屏幕（不是清除内存中的变量）。清除屏幕应用 clc 命令来完成。

（5）save 命令

当退出 MATLAB 时，在 MATLAB 工作空间中的变量会丢失。如果在退出 MATLAB 前想将工作空间中的变量保存到文件中，则可以调用 save 命令来完成，该命令的调用格式为：

save　文件名　变量列表达式　其他选项

注意：这一命令中不同的元素之间只能用空格来分隔。

例如，想把工作空间中的 A、B、C 变量存到 fileMATLAB.mat 文件中去，则可用下面的命令来实现：

```
>> save fileMATLAB A B C
```

将 A、B、C 变量存到 fileMATLAB.mat 文件中。如果想将整个工作空间中所有的变量全部存入该文件，则应采用下面的命令：

```
>> save fileMATLAB
```

当然这里的 fileMATLAB 也可省略，这时将工作空间中的所有变量自动在存入到文件 MATLAB.mat 中了。应该指出的是，这样存储的文件均是按照二进制的形式进行的，所以得出的文件往往是不可读的，如想按照 ASCII 码的格式来存储数据，则可以在命令后面加上一个控制参数－ASCII 实现。该选项将变量以单精度的 ASCII 码形式存入文件中去，如果想获得高精度的数据，则可使用控制参数－ASCII-double。

（6）load 命令

MATLAB 提供的 load 命令可以从文件中变量调出并重新装入到 MATLAB 的工作空间中去，是与 save 命令相反过程，该命令的调用格式与 save 命令相同。

当然工作空间中变量的保存和调出可单击命令窗口菜单项中“File”菜单下的“Save Worspace AS...”选项和单击“File”菜单下的“Open”选项来分别完成。

（7）exist 命令

要查看当前工作空间是否存在一个变量时，可以使用 exist 命令完成。其调用格式为：

```
A=exist('s');
```

其中，s 为要查看的变量名；

A 为返回值：

A=0 表示不存在和 s 相关的变量或文件；

A=1 表示当前工作空间存在此变量；

A=2 表示存在一个名为 s.m 的文件；

A=3 表示在当前路径下存在一个名为 s.mex 的文件；

A=4 表示存在一个名为 s..mdl 的 Simulink 文件；

A=5 表示存在一个名为 s( )的内部函数。

3．数据格式命令

MATLAB 的数据格式设置可以通过命令窗口的“File”菜单的“Preferences”选项进行，也可以通过基本命令来实现。

（1）sym 命令

sym 命令可以设置数据显示格式，并进行格式转换，以达到动态改变数据格式。其调用格式为：

sys(变量名, '参数')

式中，变量名为预设置的格式变量；参数为设置显示格式选项，见表1-2。



表1-2 sys参数设置

参数选项	数据设置格式	参数选项	数据设置格式
'd'	十进制	'e'	带上系统误差格式
'f'	浮点式	'r'	有理式

例如：

```
>> sym(pi,'d')
```

运行程序，输出结果如下：

```
ans =
3.1415926535897931159979634685442
```

（2）format 命令

format 命令用来设置输出数据格式。其调用格式如下：

format 命令参数

命令参数与功能见表1-3。

表1-3 命令参数与功能列表

参数选项	参数功能	参数选项	参数功能
format short	默认设置	format long g	15位小数
format short e	5位指数	format bank	2个十进制数
format short g	5位小数	format +	正、负或零
format long	16位整数	format rational	有理数近似
format long e	16位指数	format hex	十六进制

例如：

```
>> format long
>> pi
```

运行程序，输出结果如下：

```
ans =
   3.14159265358979
```

（3）vpa 命令

vpa 命令用来设置数据精度并计算，其调用格式如下：

S=vpa(a, b)

式中，S为返回值；a为预设定变量的变量名；b为变量的精度，并可以默认。

```
>> phi=vpa((1+sqrt(6)/3))
```

运行程序，输出结果如下：

```
phi =
1.8164965809277258124154741381062
```

1.4 MATLAB 的结构与基本运算

1.4.1 MATLAB 的结构

1. MATLAB 的数据结构

数值运算是 MATLAB 语言的显著特色，矩阵是 MATLAB 保存数据的基本形式。常用的数据量为双精度浮点类型，其 MATLAB 表示为 double。考虑到一些特殊用途，MATLAB 也引入了无符号的 8 位整型数据，其 MATLAB 表示为 uint8。

除了矩阵数据结构以外，MATLAB 还支持以下数据结构。

- 字符串型数据：MATLAB 支持字符串变量，可以用来保存相关信息，和 C 语言不同，MATLAB 字符串是用单引号引起来的，如“二阶系统的阶跃响应曲线”。
- 向量：向量分为行向量和列向量，可以看成一个 $n\times1$ 或 $1\times n$ 的矩阵。
- 数组：数组与矩阵在形式上完全一致，只是其运算与矩阵不同。矩阵运算遵守线性代数里有关矩阵运算的相关规定，而数组运算是针对每个元素。
- 元胞数组：元胞数组的基本元素是元胞，元胞可以存放任何类型数据，而且同一个元胞数组的各元胞（Cell）中的内容可以不同。元胞数组的定义符是{}，如 A={[0 1], 3; 'this is book', [2 5]}。元胞数组元素内容的访问用{}，如 A{1, 1}，结果得到[0 1]。
- 类与对象：MATLAB 允许用户自己编写各种复杂变量，如类变量。
- 符号变量：MATLAB 还定义了“符号”型变量，以区别常规的数值型变量，可以用于公式推导和数学问题的解析分析。

【例 1-3】 元胞数组的定义。

```
>> clear all;
A={[1 7;2 8],8;'12580','定义元胞数组示例'}
```

运行程序，输出结果如下：

```
A =
    [2x2 double]      [8]
    '12580'           '定义元胞数组示例'
>> A{1,1}
```

运行程序，输出结果如下：

```
ans =
     1     7
     2     8
```



```
>> A{2,2}
```

运行程序，输出结果如下：

```
ans =
```



定义元胞数组示例。

2．MATLAB 的语句结构

MATLAB 采用命令行形式的语言，每一条命令就是一条语句，其格式与书写数学表达式相近。在 Command Window 书写语句时，该语句被逐行解释执行并显示结果。如果一条语句的表达式太长，可以用“…”将其延续到下一行。如果在语句的后面加上分号“；”，则不显示执行结果。MATLAB 语句有表达式语句和赋值语句两种形式。

（1）表达式语句

表达式由变量名、常数和运算符号组成。表达式执行运算后产生的结果，将自动赋给名为“ans”的默认变量。变量 ans 的值在下一条表达式语句执行后被刷新。

【例 1-4】 表达式语句示例。

```
>> clear all;
x=3      %给变量 x 赋值为 3,这是一个赋值语句,后面讲述
x =
     3
>> x=sqrt(3)/2   %求变量 x 与 √2/2 的和
x =
    0.8660
>> x=sqrt(3)   %求变量 x 与 √2
x =
    1.7321
```

（2）赋值语句

赋值语句的格式如下：

```
变量=表达式语句
```

赋值语句执行后，将其右边表达式计算产生的结果赋值给赋值语句中等号左边的变量，并存入 MATLAB 的工作空间。MATLAB 可以同时执行以逗号“，”或分号“；”隔开的多个赋值语句。

【例 1-5】 赋值语句示例。

```
>> A=1/5
A =
    0.2000
>> X=cos(pi*A),y=sin(A)+X+1.5   %同时执行多条语句
X =
    0.8090
y =
    2.5077
>> Z=y;    %执行赋值语句,且不显示执行结果
```

```
Z    %查看变量 Z
Z =
    2.5077
```

1.4.2 MATLAB 的基本运算

1. 矩阵的实现与运算

在 MATLAB 语言中必描述矩阵的维数和类型，矩阵的维数和类型是由输入的格式和内容来确定的。例如，当 A=5 时，把 A 当做一个标量，A=1+2j 时，把 A 当做一个复数。

矩阵可以用以下方式进行赋值：

1）直接列出元素的形式。

2）通过语句和函数产生。

3）建立在文件中。

4）从外部的数据文件中装入。

（1）简单矩阵的输入

对于比较小的简单矩阵可以使用直接排列的形式输入，把矩阵的元素直接排列到方括号中，每行内的元素间用空格或逗号分开，行与行的内容用分号隔开。

例如，矩阵：

$$\boldsymbol{A}=\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}$$

在 MATLAB 下输入方式为：

```
>> A=[1 2 3;4 5 6;7 8 9]
```

运行程序，输出结果如下：

```
A =
     1     2     3
     4     5     6
     7     8     9
```

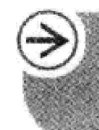
对于比较大的矩阵，可以用〈Enter〉键代替分号，对同一行的内容也可利用续行符号（…），把一行的内容分两行来输入。

例如，前面的矩阵还可以等价地由下面两种方式来输入。

```
>> A=[1 2 3;4 5 6;
      7 8 9]
```

运行程序，输出结果如下：

```
A =
     1     2     3
     4     5     6
     7     8     9
```



输入后矩阵将一直保存在工作空间中，除非被替代和清除，在 MATLAB 的命令窗口中可随时查看其内容。

利用 size 函数可测取一个矩阵的维数，该函数的调用格式为：

```
[n, m]=size(A)
```

式中，***A*** 为要测试的矩阵名，而返回的两个参数 *n*、*m* 分别为 ***A*** 矩阵的行数和列数。

当要测试的变量是一个向量时，当然仍可由 size 函数来得出的其大小；更简洁地，用户可以使用 length 函数来求出，该函数的调用格式为：

```
n=length(x)
```

式中，*x* 为要测试的向量名，而返回的 *n* 为向量 *x* 的元素个数。

如果对一个矩阵 ***A*** 用 length(***A***)函数测试，则返回该矩阵行列的最大值，即该函数等效于 max(size(a))。

例如：

```
>> size(A)
ans =
     3     3
>> n=length(A)
n =
     3
```

（2）矩阵的元素

MATLAB 的矩阵元素可用任何表达式来描述，它既可以是实数，也可以是复数。

例如：

```
>> B=[-1/4 1.3 sqrt(2) (1+2+3)*j]
```

运行程序，输出结果如下：

```
B =
  -0.2500          1.3000          1.4142          0 + 6.0000i
```

MATLAB 允许把矩阵作为元素来建立新的矩阵。对于 ***A*** 矩阵，通过下面的语句显示。

```
>> C=[A;[11 22 33]]
```

运行程序，输出结果如下：

```
C =
     1     2     3
     4     5     6
     7     8     9
    11    22    33
```

MATLAB 还允许对一个矩阵的单个元素进行赋值和操作。例如，如果想将 ***A*** 矩阵的第

3 行第 4 列的元素赋为 100，则可通过下面的语句来完成。

```
>> A(3,4)=99
```

运行程序，输出结果如下：

```
A =
     1     2     3     0
     4     5     6     0
     7     8     9    99
```

这时将只改变此元素的值，而不影响其他元素的值。

如果给出的行数或列数大于原来矩阵的范围，则 MATLAB 将自动扩展原来的矩阵，并将扩展后未赋值的矩阵元素置为 0。例如，把以上矩阵 ***A*** 的第 4 行第 5 列元素的值定义为 9，就可以通过下面语句来完成。

```
>> A(4,5)=9
```

运行程序，输出结果如下：

```
A =
     1     2     3     0     0
     4     5     6     0     0
     7     8     9    99     0
     0     0     0     0     9
```

矩阵的元素也可利用下列语句来产生：

```
S1：S2：S3
```

式中，S1 为起始值；S3 为终止值；S2 为步距。使用这样的命令就可以产生一个由 S1 开始，以步距 S2 自增，并终止于 S3 的行向量。

例如：

```
>> x=0:pi/2:2*pi
```

运行程序，输出结果如下：

```
x =
         0    1.5708    3.1416    4.7124    6.2832
```

如果 S2 省略，则可以认为自增步距为 1。

例如：

```
>> i=1:6
```

运行程序，输出结果如下：

```
i =
     1     2     3     4     5     6
```

利用上面的语句除了对单个矩阵元素进行定义之外，MATLAB 还允许对子矩阵进行定义和处理。

例如：



```
>> A(1:3,1:2:5) %表示取矩阵 A 的第 1 行到第 3 行内,且位于 1,3,5 列上的所有元素子矩阵
ans =
     1     3     0
     4     6     0
     7     9     0
>> A(2:3,:) %表示取矩阵 A 的第 2 行和第 3 行的所有元素构成的子矩阵
ans =
     4     5     6     0     0
     7     8     9    99     0
>> A(:,j)   %表示矩阵第 j 列的全部元素构成的子矩阵
>>B(:, [3,5,10])=A(:,1:3)   %表示将矩阵 A 的前 3 列，赋值给 B 矩阵的第 3、5 和 10 列
>>A(:,n:-1:1)   %表示由矩阵 A 中取 n~1 反增长的列元素组成一个新的矩阵
```

注意：A(:)在赋值语句的右边表示将 A 的所有元素按列在一个长的列向量中展开成串。

例如：

```
>> A=[1 2;3 4],B=A(:)
```

运行程序，输出结果如下：

```
A =
     1     2
     3     4
B =
     1
     3
     2
     4
```

（3）特殊矩阵的实现

在 MATLAB 中特殊矩阵可以利用函数来建立。

1）零矩阵函数 zeros。其调用格式如下：

```
A=zeros(n, m)       %产生一个 n×m 零矩阵 A
A=zeros(n)          %产生一个 n×n 零矩阵 A
A=zeros(size(B))    %产生一个与 B 矩阵同阶的零矩阵 A
```

2）单位矩阵函数 eye。其调用格式如下：

```
A=eye(n)            %产生一个 n 阶的单位矩阵 A
A=eye(size(B))      %产生与 B 矩阵同阶的单位矩阵 A
A=eye(n, m)         %0 的 n×m 矩阵
```

3）随机元素矩阵函数 rand。随机元素的矩阵的各个元素是随机产生的，如果矩阵的随

机元素满足[0, 1]区间上的均匀分布，则可以由 MATLAB 函数 rand 来生成，该函数的调用格式为：

```
A=rand(n, m)
A=rand(n)
A=rand(size(B))
```

4）全 1 矩阵函数 ones。其调用格式如下：

```
A=ones(n, m)
A=ones(n)
A=ones(size(B))
```

5）对角矩阵函数 diag。用 MATLAB 提供的方法建立一个向量 $V=[a_1,a_2,\cdots,a_n]$，则可利用 diag(V)函数来建立一个对角矩阵。

例如：

```
>> V=[2 5 3 7];A=diag(V)
```

运行程序，输出结果如下：

```
A =
     2     0     0     0
     0     5     0     0
     0     0     3     0
     0     0     0     7
```

6）上三角矩阵函数 triu 和下三角矩阵函数 tril。

调用格式为：

```
A=triu(B)
A=tril(B)
```

其中，B 为矩阵。

例如：

```
>> B=[3 7 8;1 4 7;2 6 9];
>> A=tril(B)
```

运行程序，输出结果如下：

```
A =
     3     0     0
     1     4     0
     2     6     9
```

7）伴随矩阵函数 compan。假设有一个多项式：

$$x^n+a_1x^{n-1}+\cdots+a_n$$

则可写出一个伴随矩阵：

$$P=[1 \quad a_1 \quad a_2 \quad \cdots \quad a_n]$$

其中，$P=[1 \quad a_1 \quad a_2 \quad \cdots \quad a_n]$为一个多项式向量。

例如，有一个向量$P=[1 \quad 2 \quad 3 \quad 4 \quad 5]$，则可通过下面的命令构成一个伴随矩阵。



```
>> P=[1 2 3 4 5];
>> A=compan(P)
```

运行程序，输出结果如下：

```
A =
    -2    -3    -4    -5
     1     0     0     0
     0     1     0     0
     0     0     1     0
```

2．矩阵的运算

矩阵运算是 MATLAB 的基础，MATLAB 的矩阵运算功能十分强大，并且运算的形式和一般的数学表示法相似。

（1）矩阵的转置

MATLAB 语言求矩阵 $\boldsymbol{A}$ 的转置直接用 $\boldsymbol{A}'$ 来表示。如果 $\boldsymbol{A}$ 是 $m\times n$ 的矩阵，$\boldsymbol{A}'$ 就是 $n\times m$ 的矩阵。如果 $\boldsymbol{A}$ 是复数矩阵，那么 $\boldsymbol{A}'$ 就表示复数矩阵的共轭转置。如果只想得到复数矩阵 $\boldsymbol{A}$ 的转置可以使用 $\boldsymbol{A}'$ 命令来完成，此时所得到的结果与 conj($\boldsymbol{A}'$)的结果相同。当 $\boldsymbol{A}$ 为实数矩阵时，有 $\boldsymbol{A}'=\boldsymbol{A}'$ 。

【例 1-6】 矩阵的转置示例。

```
>> A=[3 5 9;2 6 8]
A =
     3     5     9
     2     6     8
>> A1=A'
A1 =
     3     2
     5     6
     9     8
>> A2=A.'
A2 =
     3     2
     5     6
     9     8
>> B=[-2-i 4+5i;5-7i 9+2i]
B =
  -2.0000 - 1.0000i    4.0000 + 5.0000i
   5.0000 - 7.0000i    9.0000 + 2.0000i
>> B1=B'
B1 =
  -2.0000 + 1.0000i    5.0000 + 7.0000i
```

```
    4.0000 - 5.0000i     9.0000 - 2.0000i
>> B2=B.'
B2 =
  -2.0000 - 1.0000i     5.0000 - 7.0000i
   4.0000 + 5.0000i     9.0000 + 2.0000i
>> B3=conj(B')
B3 =
  -2.0000 - 1.0000i     5.0000 - 7.0000i
   4.0000 + 5.0000i     9.0000 + 2.0000i
```

（2）矩阵的加和减

矩阵的加减法的运算符为“+”和“-”矩阵只有同阶方可进行加减运算，标量可以和矩阵进行加减运算，但应对矩阵的每个元素进行加减运算。

【例 1-7】 矩阵的加减法示例。

```
>> A=[4 5 6;7 8 9;11 2 34]
A =
     4     5     6
     7     8     9
    11     2    34
>> B=[3 4 5;6 8 0;112 3 4]
B =
     3     4     5
     6     8     0
   112     3     4
>> C=A+B
C =
     7     9    11
    13    16     9
   123     5    38
>> C=A-B
C =
     1     1     1
     1     0     9
  -101    -1    30
>> D=C+10     %矩阵与标量相加
D =
    11    11    11
    11    10    19
   -91     9    40
>> A=pascal(3)     %生成 3 阶帕斯卡矩阵
A =
           1              1              1
           1              2              3
           1              3              6
>> B=magic(3)     %生成 3 阶魔方矩阵
```



```
B =
     8     1     6
     3     5     7
     4     9     2
>> X=A*B
X =
    15    15    15
    26    38    26
    41    70    39
>> Y=B*A
Y =
    15    28    47
    15    34    60
    15    28    43
```



而“.*”是向量对应元素之间的乘积，显然“.*”的条件是矩阵的类型相同。

```
>> X=A.*B
X =
     8     1     6
     3    10    21     4
     4    27    12
>> y=B.*A
y =
     8     1     6
     3    10    21
     4    27    12
```

（3）矩阵的乘法

矩阵的乘法运算符为“*”。当两个矩阵中前一矩阵的列数和后一矩阵的行数相同时，可以进行乘法运算，这与数学上的形式是一致的。

【例 1-8】 矩阵的乘法示例。

```
>> A=pascal(3)    %生成 3 阶帕斯卡矩阵
A =
     1     1     1
     1     2     3
     1     3     6
>> B=magic(3)    %生成 3 阶魔方矩阵
B =
     8     1     6
     3     5     7
     4     9     2
>> X=A*B
X =
    15    15    15
    26    38    26
```

```
    41        70        39
>> Y=B*A
Y =
    15        28        47
    15        34        60
    15        28        43
```

而“.*”是向量对应元素之间的乘积，显然“.*”的条件是矩阵的类型相同。

```
>> X=A.*B
X =
     8         1         6
     3        10        21
     4        27        12
>> y=B.*A
y =
     8         1         6
     3        10        21
     4        27        12
```

（4）矩阵的除法

矩阵的除法分左除和右除，运算符号分别为“\”和“/”，如矩阵 ***A*** 与矩阵 ***B*** 可以表示为 ***A*** \ ***B***，运算结果与矩阵 ***A*** 的逆和矩阵 ***B*** 相乘的结果相同。矩阵 ***B*** 右除矩阵 ***C*** 可表示为 ***B*** \ ***C***，运算结果与矩阵 ***B*** 和 ***C*** 的逆相乘的结果相同。

【例 1-9】 矩阵的除法示例。

```
>> A=magic(3)   %定义矩阵 A 为三维魔方阵
A =
     8     1     6
     3     5     7
     4     9     2
>> B=[1 4 7;2 5 8;3 6 9]; %定义矩阵 B
>> F=A\B           %矩阵的左除
F =
    0.0500    0.2500    0.4500
    0.3000    0.5000    0.7000
    0.0500    0.2500    0.4500
>> F=B/A     %矩阵的右除
F =
   -0.2333    1.2667   -0.2333
   -0.1667    1.3333   -0.1667
   -0.1000    1.4000   -0.1000
```

（5）矩阵的乘方

矩阵乘法运算符为“^”，如矩阵 ***A*** 的 3 次幂可写成 ***A***^3，结果为 3 个矩阵 ***A*** 相乘。

【例 1-10】 矩阵的乘方示例。

```
>> X=[1 4 7;2 5 8;3 6 9];Y=X^2,Z=Y^0.3
Y =
    30    66    102
    36    81    126
    42    96    150
Z =
   1.3646 + 0.0000i   1.4300 - 0.0000i   1.4954 + 0.0000i
   0.9388 - 0.0000i   1.6756 + 0.0000i   2.4124 - 0.0000i
   0.5130 + 0.0000i   1.9212 - 0.0000i   3.3294 + 0.0000i
```



3．矩阵的分解运算

MATLAB 拥有强大的数学处理能力，主要是因为它提供了大量的矩阵运算，这些函数能够帮助用户非常轻松地解决数学计算中那些求解过程复杂的难题。这里将要介绍一些在数值分析中占据重要的分解运算。矩阵的分解运算是指将给定的矩阵分解成特殊矩阵的乘积的过程。一般的矩阵分解运算主要有：LU（三角）分解、QR（正交）分解、CHOL（Chollesky）分解、EIG（特征值）分解和 SVD（奇异值）分解。下面将一一阐述，并用示例进行说明。

（1）QR 分解

在数值分析中，为了求解矩阵的特征值，引入 QR 分解方法。对于非奇异矩阵 $\boldsymbol{A}(n\times n)$，则存在正次矩阵 $\boldsymbol{Q}$ 和上三角矩阵 $\boldsymbol{R}$，使得 $\boldsymbol{A}=\boldsymbol{Q}*\boldsymbol{R}$，QR 分解是唯一的。MATLAB 中提供的函数为 qr。

【例 1-11】 求矩阵 ***A*** 的正交分解，其中 ***A***=[1 4 7;2 5 8;3 6 9]。

```
>> A=[1 4 7;2 5 8;3 6 9];
[Q,R]=qr(A)
Q =
   -0.2673    0.8729    0.4082
   -0.5345    0.2182   -0.8165
   -0.8018   -0.4364    0.4082
R =
   -3.7417   -8.5524  -13.3631
         0    1.9640    3.9279
         0         0    0.0000
```

（2）LU 分解

LU 分解是矩阵分解中最常见的分解方法，是方程求解方法中高斯削元法的基础，在线性方程的解法中应用非常广泛。在数值分析中，非奇异矩阵 $\boldsymbol{A}(n\times n)$，如果其顺序主子式均不为零，则存在唯一的单位下三角矩阵 $\boldsymbol{L}$ 和上三角矩阵 $\boldsymbol{U}$，使得 $\boldsymbol{A}=\boldsymbol{L}*\boldsymbol{U}$。在 MATLAB 中，提供 lu 函数来进行三角分解。

【例 1-12】 求矩阵 ***X*** 三角分解后的矩阵，其中 ***A***=[11 3 9;12 5 10;4 5 8]。

```
>> A=[11 3 9;12 5 10;4 5 8];
[L,U,P]=lu(A)
L =
```

```
    1.0000         0         0
    0.3333    1.0000         0
    0.9167   -0.4750    1.0000
U =
   12.0000    5.0000   10.0000
         0    3.3333    4.6667
         0         0    2.0500
P =
     0     1     0
     0     0     1
     1     0     0
```

（3）EIG 分解

EIG 分解利用的是 eig 函数，在做矩阵分析时，其形式上要作相应的变化。

【例 1-13】 求矩阵的特征值分解。

```
>> clear all;
A=[11 3 9;12 5 10;4 5 8];
B=magic(3);
[V,D]=eig(A,B)
V =
   -0.2917    0.6638   -0.1649
   -0.4773    0.5812    1.0000
   -1.0000   -1.0000   -0.1891
D =
    1.5482         0         0
         0   -0.4144         0
         0         0    0.3551
```

（4）CHOL 分解

矩阵 $\boldsymbol{A}(n\times n)$ 为对称正定时，则存在唯一的对角元素为正的上三角矩阵 $\boldsymbol{R}$，使得 $\boldsymbol{R}*\boldsymbol{R}'=\boldsymbol{A}$，这种分解就称为 CHOL 分解。在 MATLAB 中，提供的函数为 chol。

```
>> clear all;
A=[11 3 9;12 5 10;4 5 8];
R=chol(A)
??? Error using ==> chol
Matrix must be positive definite.
>> A=[3 -1 1;-1 5 2;1 2 4];
>> r=chol(A)
r =
    1.7321   -0.5774    0.5774
         0    2.1602    1.0801
         0         0    1.5811
```

（5）SVD 分解

奇异值分解是线性代数中一种重要的矩阵分解，在信号处理、统计学等领域有重要应

用。在 MATLAB 中，矩阵奇异值分解通过 svd 函数来实现。

【例 1-14】 求矩阵的奇异值分解。



```
>> clear all;
A=[11 3 9;12 5 10;4 5 8];
[U,S,V]=svd(A)
U =
   -0.6057   -0.3965   -0.6898
   -0.6879   -0.1747    0.7045
   -0.3999    0.9012   -0.1670
S =
   23.8040         0         0
         0    4.2069         0
         0         0    0.8188
V =
   -0.6939   -0.6783    0.2416
   -0.3048    0.5807    0.7549
   -0.6524    0.4502   -0.6097
```

1.5 多项式与数据拟合分析

1.5.1 多项式介绍

多项式在工程计算及数据处理上有着广泛的用途。在 MATLAB 中，使用行向量来表示较多的系数。向量各元素为多项式按降幂排列时的系数，默认项不得省略，如 y=x^3+4*x+12 可表示为 y=[1 0 4 12]。

【例 1-15】 已知某闭环系统特征方程为 $s^3+4s^2-s+6=0$。试用 MATLAB 表示其特征多项式，并求特征方程的特征根。

```
>> K=[1 7 2 8]    %输入多项式系数
K =
     1     7     2     8
>> roots(K)   %roots 为求多项式方程的根
ans =
  -6.8783
  -0.0608 + 1.0767i
  -0.0608 - 1.0767i
```

MATLAB 提供了一些多项式处理函数，如求多项式的值、求多项式的根、多项式的卷积分等，见表 1-4。

表 1-4 多项式处理函数

处理函数	处理函数的功能	处理函数	处理函数的功能
roots	多项式求根	conv	多项式相乘（卷积）
residue	多项式部分分式展开	deconv	多项式相除（解卷）

（续）

处理函数	处理函数的功能	处理函数	处理函数的功能
polyder	—	polyval	多项式求值
polyfit	多项式曲线拟合	polyint	多项式积分
poly	由根创建多项式	—	—

【例 1-16】 已知两个多项式 $y_1 = 2x^2 + 3x + 1$ 和 $y_2 = x + 6$，试求两个多项式的乘积，并求其在 x=4 时的值。

其实现的 MATLAB 程序代码如下：

```
>> y1=[2 3 1];        %定义多项式 y1
y2=[1 6];             %定义多项式 y2
y=conv(y1,y2)         %求多项式 y1 和 y2 的乘积 y
y =
     2    15    19     6
>> polyval(y,4)       %求当 x=4 时多项式 y 的值
ans =
   450
```

【例 1-17】 已知矩阵 $A = \begin{pmatrix} 1 & 0 & 9 \\ 7 & 2 & 6 \\ 8 & 5 & 3 \end{pmatrix}$，求矩阵 A 的特征多项式。

其实现的 MATLAB 程序代码如下：

```
>> A=[1 0 9;7 2 6;8 5 3]   %定义矩阵 A
A =
     1     0     9
     7     2     6
     8     5     3
>> P=poly(A)     %创建矩阵 A 的特征多项式
P =
    1.0000    -6.0000   -91.0000 -147.0000
```

【例 1-18】已知某特征方程的特征根为 $x_1 = 2$， $x_2 = 2 + \mathrm{i}$， $x_3 = 2 - \mathrm{i}$，求特征方程的特征多项式及其一阶导数。

其实现的 MATLAB 程序代码如下：

```
>> X=[2 2+i 2-i]   %定义由特征根组成的向量 X
X =
   2.0000                 2.0000 + 1.0000i    2.0000 - 1.0000i
>> K=poly(X)   %由特征根向量创建特征多项式
K =
     1    -6    13   -10
>> K1=polyder(P)   %多项式求导
K1 =
    3.0000   -12.0000   -91.0000
```

1.5.2 数据的插值

在已知数据中，用较简单的插值函数 $\phi(x)$ 通过所有样本点，并对临近数据进行估值计算称为插值。

插值函数 $\phi(x)$ 必须通过所有样本点。然而在有些情况下，样本点的取得本身就包含着实验中的测量误差，这一要求无疑是保留了这些测量误差的影响，满足这一要求虽然使样本点处“误差”为零，但会使非样本点处的误差变得过大，很不合理。为此，提出了另一种函数逼近方法——数据拟合法，它不要求构造的近似函数 $\phi(x)$ 全部通过样本点，而是“很好逼近”它们。

1. 一维插值

在 MATLAB 中提供 interp1 函数实现函数的一维插值。其调用格式如下：

```
Y1=interp1(x, y, X1, 'method')
%根据已知的数据（x, y），用 method 方法进行插值，然后计算 X1 对应的函数值 Y1
```

式中，x、y 是已知的数据向量，其中 x 应以升序或降序来排；X1 是插值点的自变量坐标向量；“method”是用来选择插值算法的，它可以取：“linear”（线性插值）、“cubic”（三次多项式插值）、“nearest”（最临近插值）、“spline”（三次样条插值）。

【例 1-19】 对 $y=\dfrac{1}{(2+x^2)}$，$-6\leqslant x\leqslant 6$，用 11 个结点作 3 种插值，比较结果。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x1=-6:0.3:6;
y1=1./(2+x1.^2);
x2=-6:10/(11-1):6;
y2=1./(2+x2.^2);
x=-6:0.5:6;
y3=interp1(x2,y2,x,'linear');
y4=interp1(x2,y2,x,'spline');
y5=interp1(x2,y2,x,'nearest');
subplot(221);
plot(x1,y1,'m:',x2,y2,'o');title('y=1/(2+x^2)');
subplot(222);
plot(x1,y1,'m:',x,y3);title('linear');
subplot(223);
plot(x1,y1,'m:',x,y4);title('spline');
subplot(224);
plot(x1,y1,'m:',x,y5);title('nearest');
axis([-6 6 -0.5 2]);
```



运行程序，输出效果如图 1-23 所示。

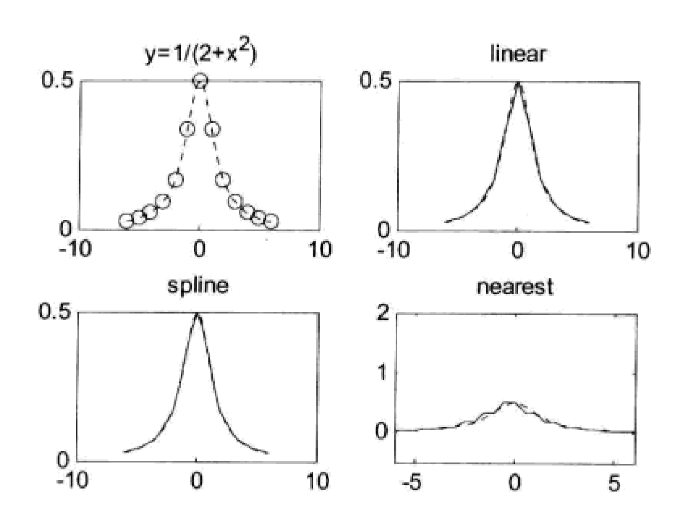


图 1-23　插值结点与不同的插值所得的插值曲线

2. 二维插值

在 MATLAB 中提供 interp2 函数实现函数的一维插值。其调用格式如下：

```
Y1=interp1(x, y, z, X1, Y1, 'method')
%根据已知的数据（x, y, z），用 method 方法进行插值，然后计算（X1, Y1）对应的函数值 Z1
```

式中，x、y 是已知的数据向量；z 是函数值；$X1$、$Y1$ 是插值的自变量坐标向量；"method"是用来选择插值算法的，它可以取："linear"（双线性插值）、"cubic"（三次多项式插值）、"nearest"（最临近插值）。

【例 1-20】 利用二维插值对 peak 函数进行插值。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
[X,Y]=meshgrid(-2:0.25:2);
Z=peaks(X,Y);
[X1,Y1]=meshgrid(-2:0.125:2);
Z1=interp2(X,Y,Z,X1,Y1);
mesh(X1,Y1,Z1)
```

运行程序，效果如图 1-24 所示。

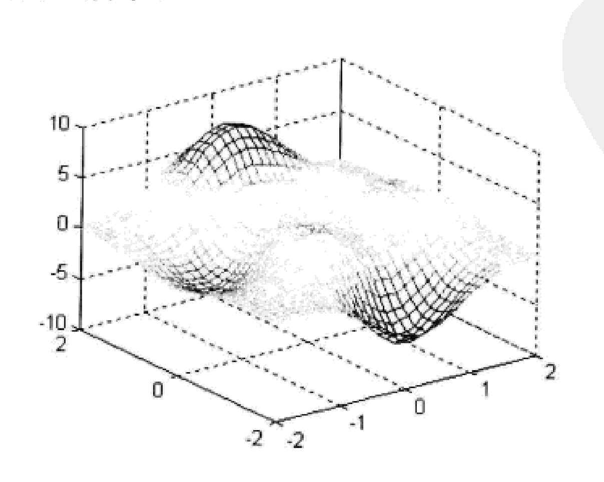
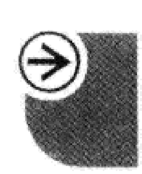


图 1-24　二维插值函数对 peak 函数进行插值



1.5.3 数据拟合分析

根据一组已知的自变量和函数的值，应用最小二乘法，求出拟合多项式并绘制拟合曲线。在 MATLAB 中提供了多项式拟合函数 polyfit。其调用格式如下：

```
p = polyfit(x,y,n)
```

式中，x、y 为数据点；n 为拟合的多项式阶次。

【例 1-21】 已知数据 x=[0 1 2 3 4 5]，y=[2 1 3 2 5 8]，求其 2 次拟合多项式和 8 次拟合多项式。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x=[0 1 2 3 4 5];   %定义 x 向量
y=[2 1 3 2 5 8];   %定义 y 向量
P2=polyfit(x,y,2);   %求 x,y 的 2 次拟合多项式
P8=polyfit(x,y,8);   %求 x,y 的 8 次拟合多项式
y2=polyval(P2,x);   %求 2 次拟合多项式对应自变量 x 的函数值
y8=polyval(P8,x);   %求 28 次拟合多项式对应自变量 x 的函数值
plot(x,y,'p')   %绘制原始数据点,数据点以“星号”显示,plot 为绘图命令
hold on   %保持绘图
plot(x,y2,'m:')   %绘制 2 次多项式拟合曲线,线型为黑色实线
plot(x,y8,'b-')   %绘制 8 次多项式拟合曲线,线型为蓝色点连线
legend('原始数据点','二次多项式拟合曲线','八次多项式拟合曲线'); %添加图例说明
xlabel('x');ylabel('y');   %添加 x,y 轴坐标
title('数据拟合曲线');   %添加标题说明
```

运行程序，输出效果如图 1-25 所示。

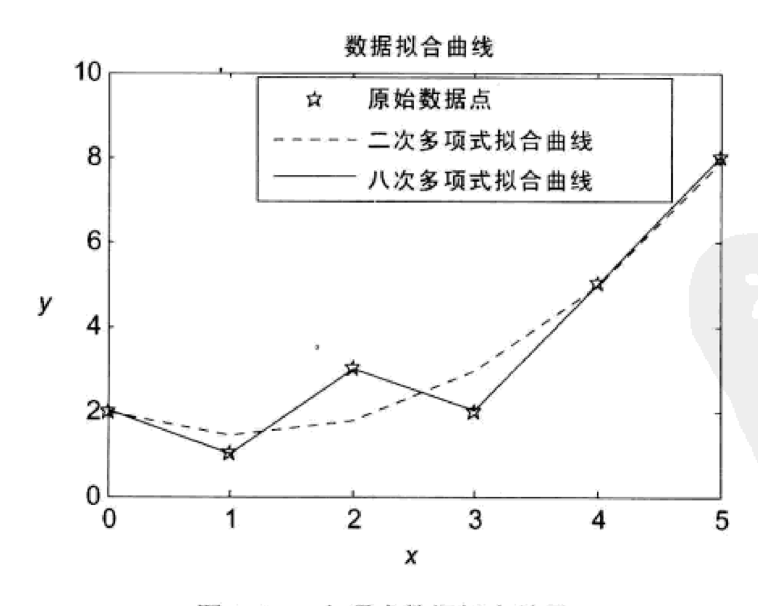


图 1-25 多项式数据拟合结果

数据拟合在工程应用研究中广泛使用。例如，我们通过检测手段，得到系统在一段时间内的输出数据，为了分析系统的性能，需要绘制系统的输出数据曲线。以前我们可以采用手

工方法，在坐标轴上用光滑曲线将数据点连成线。这种手绘图方法费时费力，误差大，不便于后期研究。

1.6 仿真的一般过程与步骤

1.6.1 仿真的一般过程

Simulink 的仿真技术则属于计算机仿真的一种。计算机仿真的一般过程如下。

1）描述仿真问题，明确仿真目的。

2）项目计划、方案设计与系统定义。根据仿真相应的结构，规定相应仿真系统的边界条件与约束条件。

3）数学建模。根据系统的先验知识、实验数据及其机理研究，按照物理原理或者采取系统辨识的方法，确定模型的类型、结构及参数。注意，在确保模型的有效性和经济性。

4）仿真建模。根据数学模型的形式、计算机类型、采用的高级语言或其他仿真工具，将数学模型转换成能在计算机上运行的程序或其他模型。

5）仿真结果分析。根据实验要求和仿真目的对实验结果进行分析处理，根据分析结果修正数学模型、仿真模型、仿真程序或者修正/改变原型系统，以进行新的实验。模型是否能够正确地表示实际系统，并不是一次完成的，而是需要比较模型和实际系统的差异，不断地修正和验证才能完成的。

1.6.2 仿真的一般步骤

系统仿真一般分为 3 个步骤，即仿真建模、仿真实验和仿真分析。应该注意的是，系统仿真是一个螺旋式发展的过程，因此这 3 个步骤可能需要循环执行多次之后才能够获得令人满意的仿真结果。

（1）仿真建模

仿真建模是根据实际系统建立仿真模型的过程，它是整个仿真过程中的一个关键步骤，因为仿真模型的好坏直接影响着仿真的结果以及仿真结构的真实性和可靠性。

仿真模型是对实际系统的一种模拟和抽象。过于简单的仿真模型会忽略实际系统的细节，在一定程度上会影响仿真结果的可靠性。但过于复杂的仿真模型则会产生很多相互因素，从而大大延长仿真时间和增加仿真结果分析的复杂度。因此，仿真模型的建立需要综合考虑其可行性和简单性。在仿真建模过程中，可以先建立一个相对简单的仿真模型，然后再根据仿真结果和仿真过程的需要逐步增加仿真模型的复杂度。

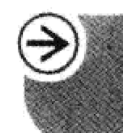
在仿真建模过程中，首先需要分析实际系统存在的问题或设立系统改造的目标，并把这些问题和目标转化成数学变量和公式。确定了仿真目标后，下一步是获取实际系统的各种运行参数，如系统占用的带宽及其频率分布、系统对于特定的输入信号产生的输出等。

在以上工作准备好以后，就是仿真软件的选择了。除了使用传统的编程语言外，目前工程技术人员比较倾向于更加专业和方便使用的专门的仿真软件。比较常见的包括 MATLAB、OPNET 和 NS2 等。

使用仿真软件建立好仿真模型后，仿真建模的这一步骤就基本完成了。值得注意的是，

在进行下一步工作前，要做好仿真模型文档说明，这有利于使仿真工作条例更加清晰，在调试过程中能够很容易找出错误所在并及时纠正。

（2）仿真实验



仿真实验是一个或一系列针对仿真模型的测试。在仿真实验过程中，通常需要多次改变仿真模型输入信号的数值，以观察和分析仿真模型对这些输入信号的反应，以及仿真系统在这个过程中表现出来的性能。需要强调的是，仿真过程中使用的输入数据必须具有一定的代表性，即能够从各种角度显著地改变仿真输出信号的数值。

在明确了仿真系统对输入/输出信号的要求之后，最好把这些设置整理成一份简单的文档。编写文档是一个好习惯，它能够帮助回忆起仿真设计过程的一些细节。当然，文档的编写不一定要求很规范，并且文档大小应该视仿真的规模而定。

对于需要较长时间的仿真，应该尽可能地使用批处理方式，使得仿真过程在完成一种参数配置的仿真之后，能够自动启动针对下一个仿真参数配置的下一次仿真。这种方式减少了仿真过程中的人工干预，提高了系统利用率和仿真效率。

（3）仿真分析

仿真分析是一个仿真流程的最后一个步骤。在仿真分析过程中，用户已经从仿真过程中获得了足够多的关于系统性能的信息，但是这些信息只是一些原始数据，一般还需要经过数值分析和处理才能够获得衡量系统性能的尺度，从而获得对仿真性能的一个总体评价。常用的系统性能尺度包括平均值、方差、标准差、最大值和最小值等，它们从不同的角度描绘了仿真系统的性能。

需要注意的是，即使仿真过程中收集的数据正确无误，由此得到的仿真结果并不一定就是准确的。造成这种结果的原因可能是输入信号恰好与仿真系统的内部特性吻合，或者输入的随机信号不具有足够的代表性。

图表是最洁的说明工具，它们具有很强的直观性，便于分析和比较，因此仿真分析的结果一般都绘成图表形式。而且，一般使用的仿真工具都具有很强的绘图功能，能够便捷地绘制各种类型的图表。

以上就是系统的一个循环。应该强调的是，仿真分析并不一定意味着仿真过程完全结束。如果仿真分析得到的结果达不到预期的目标，用户还需要重新修改仿真模型，这时候仿真分析就成为一个新循环的开始。

1.7 系统建模仿真方法与仿真工具的关系

系统建模过程是寻求系统的数学表达，即建立数学描述模型和方程的过程。仿真是对所建立模型的数值求解过程，而仿真工具则是实现这一建模与数值求解过程的软件和硬件平台。理论上，任何具有科学计算能力的计算机语言都可以作为系统仿真的软件平台。所以，仿真软件平台之间的区别不是本质的，而仅仅在于针对某类型和某层次的仿真模型在计算方便程度和用户界面上的区别。

在早期的系统设计和仿真中，由于计算机软件技术和硬件速度的限制，人们往往针对特殊的应用目的，采用当时通用计算机语言编写出相当冗长的仿真程序，然后在计算机上调试执行这些程序，得出数值结果并进行分析。仿真输出的数值结果往往是多达数十页的表格数

据，仿真程序的设计也往往不能顾及到代码的可重复使用性，所编写的仿真代码难以共享，也不能用来解决相似类型的问题，工作效率极低。

随着计算机技术的进步，在各专业领域出现了一些专用的仿真软件工具和数值计算软件包，这些仿真平台与仿真的具体问题无关，因此能够解决相关专业领域的一大类问题，对问题的建模快速而且方便。这些专用软件平台以及相应的硬件设备为用户提供了一个集成的交互式快速原型建模和仿真环境，并可以将软件模型、硬件数据以及信号综合在一起进行仿真。

在适用性上，通用的科学计算语言，如 C 语言、FORTRAN 语言等，可以作为对任何关系的仿真工具。事实上，专用的仿真平台和工具包也都是以这些通用计算机编程语言来实现的。然而，直接使用通用计算机编程语言进行系统计算和仿真需要研究人员除了具备本专业的知识基础之外，还必须具有较深的计算机程序设计知识。并且，由于程序代码的复杂性，仿真程序的调试以及仿真结果的正确性检验都是相当耗时和困难的。

严格地说，仿真平台由负责建立仿真计算机程序或仿真计算机模型的软件环境和负责仿真程序的存储、执行、数据采集和交换，以及仿真结果显示的硬件环境两部分组成。现代仿真平台和编程语言环境应具有如下基本特征。

（1）简便高效的仿真描述语言。

仿真编程的语言应当是具有一种接近于数学语言的编程描述语言，用户只需具备相应的数学知识和基本的计算机编程技能。仿真平台所提供的编程语言结构简单、便于调试验证和代码重用，这样用户可以将更多精力集中在其研究领域中，而不是消耗于琐碎的具体程序工作中。

（2）可视化的建模方法

仿真平台可提供接近于专业工程描述模型的计算机模型实现，即模型实现的直观化和可视化。在电子和通信工程中，系统模型往往除了采用数学方程描述外，还较多地采用系统方框图等图示化的方式来进行直观的描述。在计算机仿真工具中，直接对系统框图进行建模实现和仿真的平台由于物理概念清楚、直观等优点而受到工程技术人员的青睐。

（3）层次化和模块化建模的能力。

仿真平台需具有层次建模和仿真的能力。层次化建模可以轻松应付复杂系统的计算机仿真问题。在层次化模型中，一个复杂的模型可以通过多个简单的功能模块来搭建。由于简单的功能模块便于软件编码、系统测试和模块重用，从而通过层次化建模可以保证建模的效率和可靠性。例如，可以从简单的原始模块（加法器、乘法器、积分器、信号发生器等）开始，构建出滤波器、调制器等中等复杂的模块，然后再利用这些模块去构造出通信发射机和接收机，最后形成一个完整的通信系统模型。层次化的建模方法还可以使建模专注于其专业领域，无需面面俱到。比如，对于一个系统级的通信设计者来说，在层次化的建模方法下就需要关注一个滤波器模块的具体编程实现，只要知道滤波器模块的参数设置就能够构造仿真系统了。

（4）软、硬件协同仿真的能力

仿真平台要能够与相关的硬件仿真系统协同工作，完成数据采集，并将仿真结果输出到硬件系统中，从而实现硬件、软件联合的高级系统仿真和调试工作，如仿真平台上的信号处理过程能够结合数字信号处理器（DSP）的硬件实验平台等。



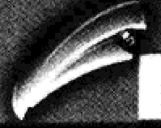
（5）交互性和图形环境

由于系统设计和分析本身就是一个交互的过程，所以也就要求仿真环境能够提供交互性，即具有交互的建模能力、交互的仿真调试能力和交互的数据分析和显示能力。作为科学研究和系统设计工具，仿真平台还应具有较强的数值可视化表现能力。作为科学研究和系统设计工具，仿真平台还应具有较强的数值可视化表现能力。科学研究成果是通过相互的学术交流来进行确认和传播的，仿真平台应便于直观地图形、曲线，甚至是“实时的”动画形式将仿真数据结果表达出来，并且能够以符合科学和出版界普遍认可的方式来进行学术交流。

（6）跨平台和可移植性

仿真软件环境应当具有较高的移植性，能够在不同的计算机硬件平台和不同的操作系统软件平台下应用。现在，随着低层语言编译器和图形化界面编程的越来越标准化，仿真平台和仿真模型程序可以在很大程度上做到与计算机硬件平台和软件平台无关，并可在各种平台上运行。





第 2 章　MATLAB 的文件结构及其绘图介绍

MATLAB 语言是一种计算机高级编程语言，提供了与 C、C++、FORTRAN 等语言的接口。而 M 文件是编辑、存储 MATLAB 程序源代码的基本形式，有着广泛的应用。此外，图形是进行数据分析、观察系统响应的重要方式。MATLAB 提供了丰富的绘图函数，既可以绘制基本二维图形，也可以绘制专业图形（如饼图、条形图、箭头图等），还可以对图形进行各种处理。

2.1　MATLAB 的程序结构

2.1.1　if 分支结构

if-else-end 语句有 3 种表达形式。

（1）单分支结构

单分支结构的语法格式如下：

```
if 表达式
    命令行
end
```

（2）双分支结构

双分支结构的语法格式如下：

```
if 表达式
    命令行 1
else
    命令行 2
end
```

（3）多分支结构

多分支结构的语法结构如下：

```
if 表达式
    命令行 1
elseif  命令行 2
    ……
else
end
```



【例 2-1】 列出 88~188 以内的所有素数，并求出它们的和。



其实现的 MATLAB 程序代码如下：

```
>> clear all;
sum=0; ss=[];
for i=88:188,
    j=0;
    for k=1:i
        if(rem(i,k)==0)   %求 i 除以 k 的余数
            j=j+1;
        end
    end
    if (j<=2)
        ss=[ss,i];
        sum=sum+i;
    end
end
disp('88 至 188 以内的素数是：');
disp(ss);
disp('88 至 188 以内的素数之和为：');
disp(sum);
```

运行程序，输出结果如下：

88 至 188 以内的素数是：

```
  Columns 1 through 12
   89    97   101   103   107   109   113   127   131   137   139   149
  Columns 13 through 19
  151   157   163   167   173   179   181
```

88 至 188 以内的素数之和为：

```
    2573
```

【例 2-2】 已知矩阵 X=[3 11 20 7 5; –39　9 17 4;–1 0 82 –5;–3 9 18 15]；求矩阵 X 中大于或等于 18、大于零且小于 18、等于零、大于–18 且小于 0，以及小于或等于–18 元素个数。

其实现的程序代码如下：

```
>> clear all;
X=[3 11 20 7; -39   9 17 4;-1 0 82 -5;-3 9 18 15];
[m,n]=size(X);   %求矩阵 X 的维数
ss1=[];ss2=[];
ss3=[];ss4=[];
ss5=[];
for i=1:m*n
    if X(i)>=18                 %if 单分支结构
        ss1=[ss1,X(i)];         %将矩阵 X 中大于 18 的元素放在向量 ss1 中
        X(i)=20;                %将矩阵 X 中大于 18 的元素重新赋值 20
    end
```

```
        if (X(i)<18)&(X(i)>0)           %if 多分支结构
            ss2=[ss2,X(i)];             %将矩阵 X 中大于 0 且小于 18 的元素放在向量 ss2 中
            X(i)=10;                    %将矩阵 X 中大于 0 且小于 18 的元素重新赋值 10
        elseif X(i)==0
            ss3=[ss3,X(i)];             %将矩阵 X 中等于 0 的元素放在向量 ss3 中
        elseif (X(i)<0) &(X(i)>-18)
            ss4=[ss4,X(i)];
                X(i)=-5;
        else
            ss5=[ss5,X(i)];
            X(i)=-18;
        end
    end
    sum1=length(ss1);   %求向量 ss1 的长度
    sum2=length(ss2);   %求向量 ss2 的长度
    sum3=length(ss3);   %求向量 ss3 的长度
    sum4=length(ss4);   %求向量 ss4 的长度
    sum5=length(ss5);   %求向量 ss5 的长度
    disp('矩阵 X 中大于或等于 18 的元素个数='),sum1
    disp('矩阵 X 中大于或零小于 18 的元素个数='),sum2
    disp('矩阵 X 中等于元素个数='),sum3
    disp('矩阵 X 中大于-18 且小于 18 的元素个数='),sum4
    disp('矩阵 X 中小于或等于-18 的元素个数='),sum5
```

运行程序，输出结果如下：

```
矩阵 X 中大于或等于 18 的元素个数=

    sum1 =
        3

矩阵 X 中大于 0 且小于 18 的元素个数=

    sum2 =
        8

矩阵 X 中等于元素个数=

    sum3 =
        1

矩阵 X 中大于–18 且小于 18 的元素个数=

    sum4 =
        3

矩阵 X 中小于或等于–18 的元素个数=

    sum5 =
        4
```

2.1.2 循环结构

MATLAB 的程序循环结构有两种控制方法：一种是采用 while 循环结构；一种是采用

for 循环结构。

（1）while 循环结构

while 循环结构的语句格式如下：

```
while 表达式
    执行指令语句
end
```



当表达式值为真时，执行循环体内的指令语句；当表达式为假时，循环结束。在使用 while 循环时，一定要注意避免表达式为恒真的情况，否则程序将陷入死循环。

【例 2-3】 已知某向量元素间满足 $a_{k+3}=a_{k+2}+a_{k+1}+a_k$，且已知 $a_1=1$，$a_2=2$，$a_3=3$，用 while 循环求该向量中第一个大于 999 的元素。

其实现的 MATLAB 程序代码如下：

```
>> A=[];  %定义向量 a 为空向量
A(1)=1;A(2)=2;
A(3)=3;
i=3;                %定义循环起始的向量下标
while A(i)<=999
    i=i+1;          %若省略此句，则程序陷入死循环
    A(i)=A(i-1)+A(i-2)+A(i-3);
end
disp('第一个大于 999 的元素的下标是');
i
disp('该元素的数值是');
A(i)
```

运行程序，输出结果如下：

第一个大于 999 的元素的下标是

```
i =
    13
```

该元素的数值是

```
ans =
        1431
```

（2）for 循环结构

for 循环结构的语法格式如下：

```
for i=表达式
    表达式语句,……,表达式语句
end
```

i 为循环指针变量，表达式通常是一个向量，如 i=m:s:n，m 为初始值，n 为终止值，s 为增量（也即步长）。如果 s 为正值，要求 m<n，如果 s 为负值，要求 m>n。当循环指针变量 i 的值等于向量的某个元素时，执行循环体内的表达式语句，如果循环指标变量 i 的值超

过向量的上限，则循环结束。

【例 2-4】 用 for 循环求 $\prod_{i=1}^{12} x$ 的值。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
os=1;
for i=1:12   %增量为 1 可省略
      os=os*i;
end
os   %显示乘积结果
```

运行程序，输出结果如下：

```
os =
   479001600
```

2.1.3 switch 分支结构

switch-case 分支结构的语法格式如下：

```
switch sig   %sig 为一标量或字符串
     case 1   %当 sig 等于 1 时，执行命令 1，以下相同
          命令行 1
     case 2
          命令行 2
          ……
     case n
          命令行 n
     otherwise
          命令行 n+1
end
```

【例 2-5】 试绘制二阶系统 $y=1+\frac{e^{-\xi\omega_n t}}{\beta}\sin(\omega_n\beta t+\theta)$ 的单位阶跃响应图形，其中 $\beta=\sqrt{1-\xi^2}$，$\theta=\arctan\sqrt{\frac{1-\xi^2}{\xi}}$，$\omega_n=0.2$，$\xi=0.25$、0.5、0.75，时间范围 1~80s。

程序如下：

```
>> clear all;
t=0:0.1:80;      %定义时间向量
for ksi=0.3:0.3:0.9
     be=sqrt(1-ksi^2);
     bi=atan(be/ksi);
     wn=0.2;
     y=1-(exp(-ksi*wn.*t)./be).*sin((be*wn).*t+bi);
     switch round(ksi*10)            %根据不同的 be，曲线采用不同的颜色
          case 3
               plot(t,y,'r:');            %当 be=0.25 时，用黑色点画线
```

```
            hold on
        case 6
            plot(t,y,'m-');           %当 be=0.5 时，用紫色实线
        case 9                        %当 be=0.75 时，用蓝色虚线
            plot(t,y,'b--');
        otherwise
            disp('输入错误');
    end
end
legend('ke=0.3','ke=0.6','ke=0.9');   %添加图例
title('二阶系统阶跃输入响应');          %添加标题
xlabel('时间/秒');                     %添加 x 坐标
ylabel('输出响应');                    %添加 y 坐标
grid on
```

运行程序，输出效果如图 2-1 所示。

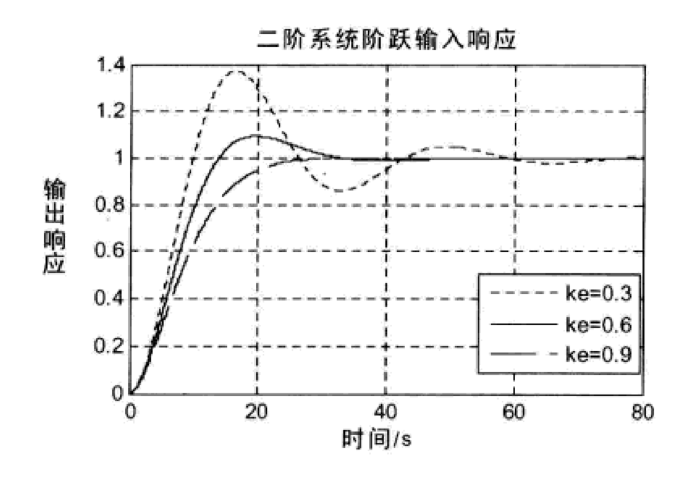


图 2-1 二阶系统单位阶跃响应曲线



2.2 M 文件

本节将介绍 M 文件的基本类型，主要包括数据文件、函数式 M 文件，以及脚本式 M 文件。

2.2.1 数据文件

数据文件是 MATLAB 中经常使用的用于保存变量的文件，该文件的扩展名为 mat。数据文件是以标准二进制格式将变量进行保存的一种文件格式，使用数据文件可将工作空间中的全部或部分数据变量保存下来。数据文件的生成和调用是由 save 函数和 load 函数完成。

【例 2-6】 数据文件的调用。

其实现的 MATLAB 程序代码如下：

```
>> load gatlin    %gatlin 是 MATLAB 自带的一个图片数据文件
image(X)
```

运行程序，输出效果如图 2-2 所示。

更改图形窗口的色图，并刷新图形显示。在命令窗口中输入以下内容：

```
>> colormap(gray)
```

运行程序，输出效果如图 2-3 所示。

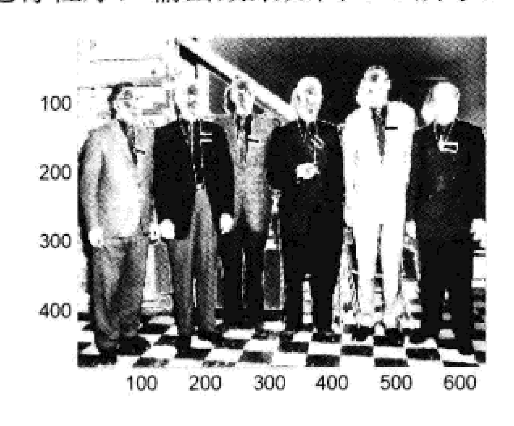


图 2-2　M 文件的调用（一）

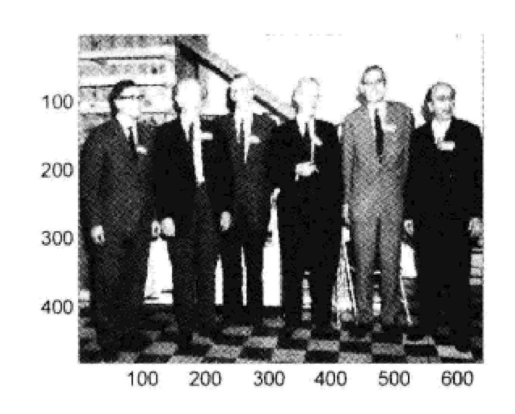


图 2-3　M 文件的调用（二）

2.2.2 M 文件简介

M 文件的编辑语法类似于 C 语言，但又有其自身特点。M 文件只是一个简单的 ASCII 码文件，执行程序时逐行解释运行程序。M 文件可分为两种类型：命令脚本文件和函数文件，其中函数文件又可以分为内置函数、自定义函数、S-函数等各种类型。

1. 命令脚本文件

命令脚本文件实际上是一串指令的集合。命令脚本文件的执行结果与在命令窗口逐行执行文件中的所有指令性计划的结果是一样的。命令脚本文件没有输入、输出参数。文件运行后，产生的所有变量都保存在 MATLAB 的基本工作空间中，除非用户使用 clear 指令清除或关闭 MATLAB，否则这些变量将一直保存在基本工作空间中。

命令脚本文件包括两部分：注释部分和程序部分。其中，注释部分必须在符号“%”之后，MATLAB 不对其进行计算，只供程序设计人员和阅读者方便理解程序而用。程序部分就是程序中一般的命令行和程序段，MATLAB 要对其进行编译和计算。

【例 2-7】 用 M 命令脚本文件画出衰减振荡曲线 $y = e^{-\frac{t}{4}\cos 4t}$ 及其他的包络线 $y_0 = e^{-\frac{t}{4}}$。t 的取值范围是$[0,\ 5\pi]$。

其实现步骤如下：

1）打开 MATLAB 命令窗口，单击“File”菜单下“New”选项中的“Mfile”命令，打开编辑窗口。

2）在编辑窗口逐行编写以下语句：

```
>> clear all;
t=0:pi/50:5*pi;
y0=exp(-t/4);
y=exp[(-t/4).*sin(4*t)];
plot(t,y,'-r',t,y0,':b',t,-y0,':b')
```

3）保存 M 文件，并且保存在搜索路径上，文件名为 li2_7.m。

4）运行 M 文件。在命令窗口输入 li2_7.m，并按〈Enter〉键，或者在编辑窗口打开 Tools 菜单，再选择“Run”命令。在 Figure 图形窗口出现曲线如图 2-4 所示。

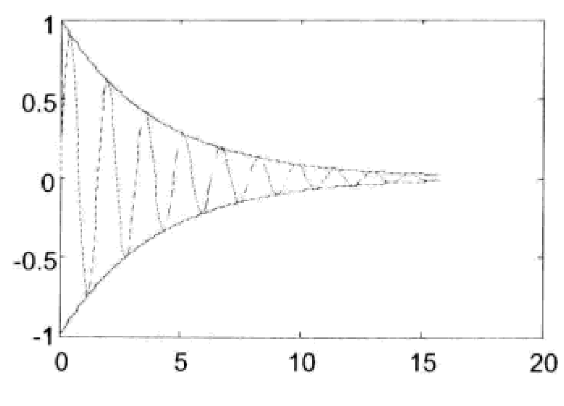


图 2-4 衰减振荡曲线与包络



M 命令文件中的语句可以访问 MATLAB 工作空间中的所有变量与数据，同时 M 命令文件中的所有变量都是全局变量，可以被其他的命令文件与函数文件访问，并且这些全局变量一直保存在内存中，可以用 clear 命令来清除这些全局变量。

2．MATLAB 内置函数文件

MATLAB 自定义的函数文件称内置函数文件。调用内置函数的方法是使用函数名并给出相应的入口、出口参数即可。

【例 2-8】 MATLAB 内置函数的调用。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
x=0:pi/10:pi;
```

生成变量 x，下面调用内置 sin 函数和 plot 绘图函数。在命令窗口输入以下内容：

```
>> y=sin(x);
```

在命令窗口中输入以下内容：

```
>> plot(x,y);
```

运行程序后，生成如图 2-5 所示的效果图。

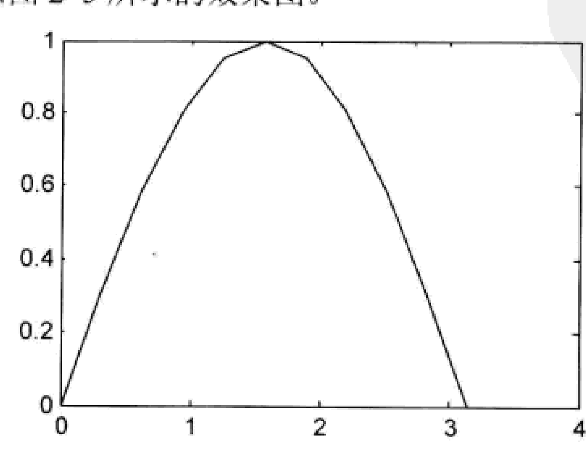


图 2-5 MATLAB 内置函数的调用

3. 用户自定义的 M 函数文件

MATLAB 用户可以根据需要编辑自己的 M 函数文件，它们可以像 MATLAB 提供的库函数一样被方便地调用。这种调用 MATLAB 语言创建与定义新函数的功能，体现了 MATLAB 语言强大的扩展功能。

用户自定义函数文件编写时需要有输入变量和输出变量。M 函数文件的一般格式为：

```
function 返回变量=函数名（输入变量）
%注释说明语句段
程序语句段
```

M 函数文件第一行必须以关键字 function 作为引导词，M 函数文件的文件名必须是“<函数名>.m”。程序中的变量均为局部变量，不保存在工作空间中，变量只在函数运行期间有效。

【例 2-9】 设可逆方阵为 $\boldsymbol{A}$，给定同时求 $|\boldsymbol{A}|$， $\boldsymbol{A}^2$， $\boldsymbol{A}^{-1}$， $\boldsymbol{A}'$ 的 M 函数文件。

1）单击“File”菜单下“New”选项中的“Mfile”命令，打开编辑窗口。

2）在编辑窗口输入以下语句：

```
function [d1,d2,inva,traa]=comp4(x)
% M 函数文件 comp4.m 同时求矩阵 x 的 4 个值
% d1 为矩阵 x 的行列式
% d2 为矩阵 x 的平方
% inva 为矩阵 x 的逆矩阵
% traa 为矩阵 x 的转置
d1=det(x)
d2=x^2
inva=inv(x)
traa=x'
```

3）保存 M 函数文件，并且保存在搜索路径上，文件名为 comp4.m。

4）在命令窗口运行以下代码：

```
>> x=[1 2;5 8];            %输入矩阵
>> comp4(x);               %调用 comp4.m 函数计算矩阵 x 的 4 个值
```

运行程序，输出结果如下：

```
d1 =
     -2
d2 =
     11     18
     45     74
inva =
    -4.0000     1.0000
     2.5000    -0.5000
traa =
      1      5
```

```
    2     8
```

说明：1）第一行执行命令的作用是以指明该 M 文件为 M 函数文件。其中自变量为 x，因变量为 d1、d2、inva、traa，因为是多个因变量故用“[ ]”括起来，comp4 是函数名。自变量与因变量既可以是数值，也可以是字符串。



2）变量 x 对于 M 函数文件 comp4.m 是局部变量，当函数调用结束后，变量 x 不再存在。

3）在 M 文件前面，连续几行带符号“%”的注释行有两个作用：一是随 M 文件全部显示与打印时，直接起解释提示作用；二是供 help 指令与 lookfor 指令联机查询使用。

【例 2-10】 在线查询例 2-9 的函数 comp4.m 的使用说明。

```
>> help comp4
```

运行程序，输出结果如下：

```
  M 函数文件 comp4.m 同时求矩阵 x 的 4 个值
  d1 为矩阵 x 的行列式
  d2 为矩阵 x 的平方
  inva 为矩阵 x 的逆矩阵
  traa 为矩阵 x 的转置
>> lookfor comp4
```

运行程序，输出结果如下：

```
comp4.m: % M 函数文件 comp4.m 同时求矩阵 x 的 4 个值
```

4．系统文件 S-函数

系统文件 S-函数用于描述系统运动的专用函数，是特殊的 M 文件。创建 S-函数的方法有 3 种：一是由 Simulink 结构图自动创建；二是用函数 M 文件编写；三是采用 C 语言编写 mex 文件直接定义。S-函数一旦创建，既可在框图中使用，也可在文件中调用。S-函数的一般调用格式如下：

```
[sys, x0]=sfunction(t, x, u, flag)
```

其中，sys 是系统状态；x0 是状态初值；sfunction 表示用户定义的系统；t 是当前时刻；x 是当前状态；u 是当前输入值；flag 是标志量。

S-函数与函数 M 文件类似，只是输入、输出变量是限定的。

5．函数句柄

函数句柄是 MATLAB 6 之后特有的语言结构，优点是方便实现函数间互相调用，兼容函数加载的所有方式，拓宽子函数包括局部函数的使用范围，提高函数调用的可靠性，减少程序设计中的冗余，提高重复执行的效率，数组、结构数组、细胞型数组能够结合定义数据。

定义数据句柄只需在提示符@后添加相应函数的函数名，函数句柄的内容通过 functions 函数显示。调用可通过 feval 函数进行，调用的格式如下：

```
feval(函数句柄，参数列表)
```

函数句柄与函数名字符串转换可通过以下命令实现：

```
函数句柄名 str2func('函数名字符串')      %函数名字符串转换函数句柄
func2str(函数句柄名)                     %函数句柄转换函数名字符串
```

【例 2-11】 函数句柄的创建和显示。

在命令窗口输入以下代码：

```
>> f_h=@sym
```

创建的函数句柄如下：

```
f_h =
    @sym
```

在命令窗口中输入以下代码：

```
>> functions(f_h)
ans =
    function: 'sym'
        type: 'simple'
        file: 'F:\matlab\toolbox\symbolic\@sym\sym.m'
```

【例 2-12】 函数句柄的调用。

首先创建函数句柄 f，在命令窗口输入以下代码：

```
>> f=@rand
f =
    @rand
>> functions(f)
ans =
    function: 'rand'
        type: 'simple'
        file: 'MATLAB built-in function'
```

在命令窗口输入以下代码：

```
>> feval(f,6)
```

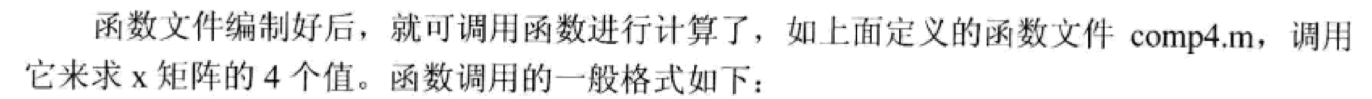
运行程序，输出结果如下：

```
ans =
    0.9501    0.4565    0.9218    0.4103    0.1389    0.0153
    0.2311    0.0185    0.7382    0.8936    0.2028    0.7468
    0.6068    0.8214    0.1763    0.0579    0.1987    0.4451
    0.4860    0.4447    0.4057    0.3529    0.6038    0.9318
    0.8913    0.6154    0.9355    0.8132    0.2722    0.4660
    0.7621    0.7919    0.9169    0.0099    0.1988    0.4186
```

2.3 函数文件的分析

2.3.1 调用函数





函数文件编制好后，就可调用函数进行计算了，如上面定义的函数文件 comp4.m，调用它来求 x 矩阵的 4 个值。函数调用的一般格式如下：

```
[输出实参数]=函数名(输入实参表)
```

需要注意的是，函数调用时各实参出现的顺序和个数，应与函数定义时形参的顺序、个数一致，否则会出错。函数调用时，先将实参传递给相应的形参，从而实现参数传递，然后再执行函数的功能。

【例 2-13】 利用函数文件，实现直角坐标（x,y）与极坐标(ρ,θ)之间的转换。

已知转换公式为：

极坐标的矢径：$\rho=\sqrt{x^2+y^2}$

极坐标的极角：$\theta=\arctan\left(\frac{y}{x}\right)$

函数文件 xtran.m：

```
function [rh,th]=xtran(x,y)          %直角坐标与极坐标之间的转换函数

rh=sqrt(x*x+y*y);
th=atan(y/x);
```

其调用的程序代码如下：

```
>> x=input('Please input x=:');
y=input('Please input y=:');
[rh,th]=xtran(x,y);
rh
th
```

运行程序，输入如下代码：

```
Please input x=:2
Please input y=:3
```

在命令窗口输出结果如下：

```
rh =
     3.6056
th =
     0.9828
```

在 MATLAB 中，函数可以嵌套调用，即一个函数可以调用别的函数，甚至调用它自身。一个函数调用它自身称为函数的递归调用。

【例 2-14】 利用函数的递归调用，求 $m!$。

$m!$本身就是以递归的形式定义的：

$$m! = \begin{cases} 1 & m \leqslant 1 \\ m\ (m-1)\ ! & n > 1 \end{cases}$$

显然，求 $m!$需要求（m–1）!，这时可采用递归调用。递归调用函数文件 xfactor.m 源代码如下：

```
function f=xfactor(m)
if m<=1
    f=1;
else
    f=xfactor(m-1)*m;     %递归调用求(m–1)!
end
```

如果求 s=1!+2!+3!+4!+5!+6!。即实现调用的程序代码如下：

```
>> s=0;
for i=1:6;
    s=s+xfactor(i);
end
s
```

运行程序，输出结果如下：

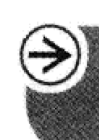
```
s =
   873
```

【例 2-15】 任意排列问题。

MATLAB 提供的 randperm(n) 函数，可以产生一个从整数 1～n 的任意排列。编写一个函数来实现 randperm(n)函数的功能，即给出一个由任意数组组成的行向量，然后产生这个行向量元素的任意排列。

考虑编写两个不同的函数，这两个函数的功能完全相同，只是控制结构不同。第一个函数用循环结构，可以用 for 语句或 while 语句控制循环，第二个使用函数的递归调用。

```
%第一个函数用循环结构
function Y=xrandpA(X)
% xrandpA 用 for 循环产生一个行向量的任意排列
% xrandpA(X)产生行向量 x 的任意排列
[m,n]=size(X);
if m>1
    error('xrandpA accepts as inputs only vector');
end
Y=[];
l=n;
for i=1:n
    k=1+fix(l*rand);
```



```
        x=X(k);
        Y=[Y,x];
        X(k)=[];
        l=l-1;
    end
    %第二个函数用函数的递归调用
    function Y=xrandpB(X)
    % xrandpB 用 for 循环产生一个行向量的任意排列
    % xrandpB(X)产生行向量 x 的任意排列
    [m,n]=size(X);
    i=n;
    if m>1
        error('xrandpB accepts as inputs only vectors');
    end
    if n<=1
        Y=X;
    else
        k=1+fix(j*rand);        %随机选择 y 的下一个元素的位置
         x=X(k);                %被选择的元素
         X(k)=[];               %从 x 中删除 X 元素
         z=xrandpB(X);          %将剩下的元素随机排列
         Y=[z,x];               %构造输出向量
         i=i-1;
    end
```

在命令窗口调用所定义的函数。例如：

```
>> xrandpA([3 9 54 2 78 -3 458 1 7])
```

递归调用后，输出结果如下：

```
ans =
     3    458     2     1    78    -3    54     9     7
```

因为 MATLAB 将长度为 *n* 的字符串当做一个长度为 *n* 的向量，所以也可以用字符串作为函数的自变量。例如：

```
>> xrandpB('elephat')
```

运行程序，输出结果如下：

```
ans =
tahpele
```

显然，此时函数可以用于做字谜游戏。

2.3.2 函数的参数

调用函数时，总会有一些数据作为输入参数传给被调用的函数。特别强调的是，MATLAB 总是按值传送输入参数。

函数结束时，也可能有处理结果返回给调用函数。在函数定义中，返回给调用函数的数据称为输出参数。

对于输入/输出参数，MATLAB 提供了一些函数来处理它们。

1．查询输入/输出参数的个数

（1）nargin 函数和 nargout 函数

nargin 函数和 nargout 函数的使用，能够在调用一个函数时，确定函数有几个输入/输出参数。

其调用格式如下：

```
n = nargin
n = nargout
n = nargin(fun)
n = nargout(fun)
```

fun 为被测定函数的函数名或函数句柄。

上面的两种格式，用在 M 文件函数体内，旨在查询调用这个函数时指定的输入（nargin）或输出（nargout）的参数个数。

下面的两种格式，用在被测定的 M 文件函数体外，旨在得到名为 fun 的函数定义时声明的输入或输出的参数个数。

如果被测定函数的输入或输出参数的数目是不确定的，则 n 是负数。

（2）查询参数个数的意义

对一些函数，并不知道规定的输入或输出参数的个数，因此调用它们时就不能确定要提供多少输入或输出参数，有必要测试一下，以免调用时出错。

另外，在被调用的函数中，通过测定用户提供的参数个数，确定程序走向，对不同的参数个数，做不同的处理。

【例 2-16】 xiuplot 函数的部分源代码。

```
function [x1,y1]=xiuplot(x,y,np,an,su)
% xiuplot 为绘图函数
% 调用格式 xiuplot(x,y,np,an,su)
% 前两个参数是用户输入的，后 3 个参数是默认值
……
if nargin<5,
    su=20;
end
if nargin<4,
    an=10;
end
if nargin<3,
    np=25;
end
……
if nargout==0
    plot(x,y)
```



```
else
    x1=x;
    y1=y;
end
```



【例 2-17】 函数在输入的字串中寻找第一个 xto。一个 xto 是由空线间隔（while space）或其他字符定界的字串。

```
function [to,re]=xstrto(str,deter)
% 此函数至少要输入一个参数
if nargin<1
    error('Not enough input arguments.');
end
to=[];re=[];
len=length(str);
if len==0
    return
end
% 假如输入一个参数，要使用空格键
if (nargin==1)
    deter=[8:12 30];
end
i=1;
% 阻止非字符的输入
while (any(str(i)==deter))
    i=i+1;
    if (i>len),
        return;
    end
end
% 结束程序
st=i;
while(~any(str(i)==deter))
    i=i+1;
    if (i>len),
        break;
    end
end
finish=i-1;
to=str(st:finish);
% 输出两个参数，空格后开始计数
% first deter(re)
if(nargout==2)
    re=str(finish+1:end);
end
```

用一个输入参数调用这个函数，输出结果如下：

```
>> [to,re]=xstrto('str deter')
to =
str deter
re =
    Empty string: 1-by-0
```

用两个输入参数调用这个函数，输出结果如下：

```
>> [to,re]=xstrto('strAAB deterCCD',',')
to =
strAAB deterCCD
re =
    Empty string: 1-by-0
```

2．传递或返回可变数目的参数

有一些函数，输入的参数或返回到调用函数的输出参数不确定，可多可少。对这样的输入/输出参数，MATLAB 要作特别的处理。这里涉及两个函数 varargin 和 varargout。

（1）varargin 函数

varargin 函数调用格式如下：

```
function    y=bar(varargin)
```

bar 函数接受任意数目的输入参数，而 MATLAB 将这些参数构成一个单元数组，varargin 则为单元数组名。

【例 2-18】 testvar1 函数接受任意数目的向量，每个向量有两个元素，表示坐标上的一个点，然后在绘图窗口把这些点连成线。

```
function testvar1(varargin)
for k=1:length(varargin)
    x(k)=varargin{k}(1);
    y(k)=varargin{k}(2);
end
xmin=min(0,min(x));
ymin=min(0,min(y))'
axis([xmin fix(max(x))+3 ymin fix(max(y))+3])
plot(x,y)
```

对 y(k)=varargin{k}(2);语句作说明如下：

单元数组的索引由两个下标组成：{k}指明要取哪个单元，（2）指明取单元中的哪个元素。

用两组不同数目的参数调用：

```
>> testvar1([1 7],[2 5],[3 6],[7 9],[5 8],[0 10])
```

运行程序，效果如图 2-6 所示。

```
>> testvar1([-1 0],[2 -5],[3 6],[2 1])
```

运行程序，效果如图 2-7 所示。

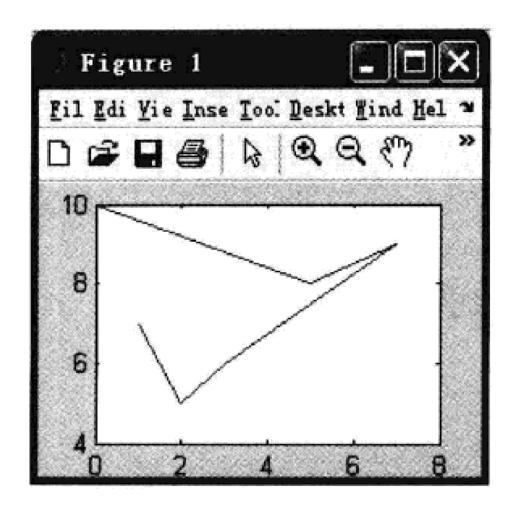


图 2-6 绘图窗口（一）

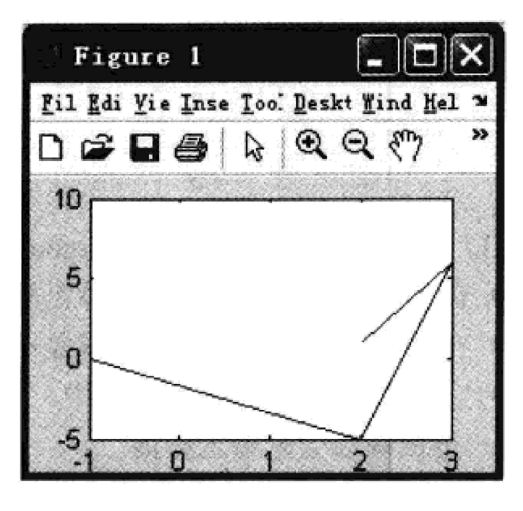


图 2-7 绘图窗口（二）



（2）varargout 函数

varargout 函数调用格式如下：

```
function varargout=foo(n)
```

从函数 foo 返回任意个输出参数，返回的参数包含在 varargout 函数中。与 varargin 函数相反，返回输出的函数必须把单个的数据组合成单元数组，放在 varargout 函数（它又变成了单元数组名）中，以便于 MATLAB 把输出参数返回给调用函数。而输出时，MATLAB 则将单元数组拆成单个数据。

【例 2-19】 varargout 函数示例。

```
function [varargout]=testvar2(xin)
for k=1:nargout
    varargout{k}=xin(k,:);
end
```

xin 如例 2-18 一样，输入若干个成对元素的向量。函数很简单，将输入参数直接输出，主要演示如何把单个的输出数据组合成单元数组。

varargout{k}=xin(k,:); 语句为单元数组赋值。等号左边是单元数组，花括号及其中的 k 表示单元数组的一个单元。

调用 testvar2 函数：

```
>> X=[0 1 1;1 2 2 ;2 3 3;3 4 4;4 5 5];
>> [x1,x2,x3,x4,x5]=testvar2(X)
```

运行程序，输出结果如下：

```
x1 =
     0     1     1
x2 =
     1     2     2
```

```
x3 =
      2     3     3
x4 =
      3     4     4
x5 =
      4     5     5
```

（3）使用 varargin 函数和 varargout 函数的原则

- 在函数定义行中，它们必须是输入/输出参数列表中的最后一个参数。
- 在函数定义行中，它们必须是小写字母。
- 它们只被用在 M 文件函数中。

3．返回被改变的输入参数

有这种情况：作为输入参数的变量，其值在被调用的函数中改变了。可是下次调用这个函数时，要用它的当前值，怎么办呢（因为被调用的函数结束时，变量已从内存中清除了，没有当前值）？

比如 readText 函数，每次被调用都从文件上读一行文本，并且要记住被读文件的位置，以便下次接着读；否则，每次都重复读一行。假如位置变量是 offset，为了保持 offset 的当前值，一个办法是在调用函数的输入/输出参数列表中都包含 offset 变量，即

```
[text, offset]=readText(filestart, offset)
```

如此，本次调用的输出 offset，作为下次调用的输入 offset。

2.3.3 局部变量与全局变量

在 MATLAB 中，函数文件中的变量是局部的，与其他函数文件及 MATLAB 工作空间相互隔离，即在一个函数文件中定义的变量不能被另一个函数文件引用。如果在若干函数中，都把某一变量定义为全局变量，那么这些函数将共用这个变量。全局变量的作用域是整个 MATLAB 工作空间，即全程有效，所有的函数都可以对它进行存取和修改。因此，定义全局变量是函数间传递信息的一种手段。

全局变量用 global 命令定义，其一般格式如下：

```
global 变量名
```

【例 2-20】 全局变量示例。

建立函数文件 xiu2_20.m，该函数将输入的参数加权相加：

```
function f=xiu2_20(x,y)
global aa bb
f=aa*x+bb*y;
```

在命令窗口输入以下代码：

```
>> global aa bb
aa=3;
bb=5;
```



```
s=xiu2_20(3,5)
```

运行程序，输出结果如下：

```
s =
    34
```



由于在 xiu2_20 函数和基本工作空间中都把 aa 和 bb 两个变量定义为全局变量，所以只在命令窗口中改变 aa 和 bb 的值，就可改变加权值，而无需修改 xiu2_20.m 文件。

在实际编程时，可在所有需要调用全局变量的函数定义全局变量，这样就可实现数据共享。为了在基本工作空间中使用全局变量，也要定义全局变量。在函数文件里，全局变量的定义语句应放在变量使用之前，为了便于了解所有的全局变量，一般把全局变量的定义语句放在文件的最前部。

值得注意的是，在程序设计中，全局变量固然可以带来方便，但却破坏了函数对变量的封装，降低了程序的可读性。因而，在结构化程序设计中，全局变量是不受欢迎的。尤其当程序较大、子程序较多时，全局变量将给程序调试和维护带来不便，故不提倡使用全局变量。如果一定要用全局变量，最后给它起一个能反应变量含义的名字，以免和其他变量混淆。

2.4 MATLAB 的绘图功能

MATLAB 为控制界广泛接受的另一个主要原因是它提供了十分方便的一系列绘图命令，包括线性坐标、对数坐标、半对数坐标及极坐标等命令，允许用户同时打开若干个图形窗口并对图形进行标注文字说明等，使得图形绘制和处理复杂工作变得简单。

2.4.1 二维图形绘制

1. 基本形式

MATLAB 最基本的绘图函数为 plot，如 y 是一个 n 维向量，那么 plot(y)绘制一个 y 元素和 y 元素排列序号 1，2，…，n 之间关系的线性坐标图。

例如：

```
>> y=[0.01 0.48 0.9 1.2 2.3 4.5 0.24 0.01];
plot(y)
```

运行程序，显示效果如图 2-8 所示。

2. 多条线型

在同一图形中可以绘制多条线型，其调用格式如下：

```
plot(X1,Y1, X2,Y2,…)
```

以上命令可将 X1 对 Y1，X2 对 Y2,…，的图形绘制在一个图形中，而且分别采用不同的色彩或线型，以下命令可输出多条曲线。

```
>> x=0:0.12:2*pi;
```

```
>> plot(x,cos(x),x,sin(x))
```

运行程序，显示效果如图 2-9 所示。

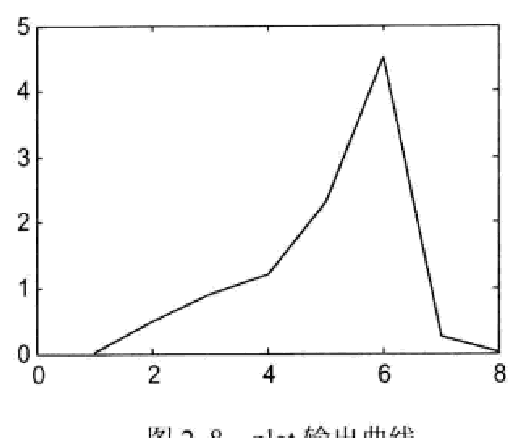


图 2-8 plot 输出曲线

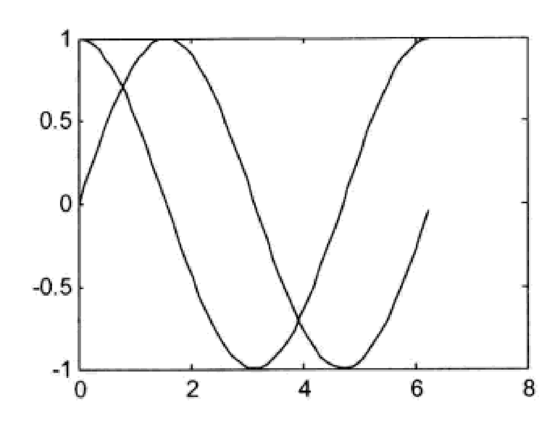


图 2-9 plot 输出多条曲线

当 plot 命令用于复数数据时，通常虚部是忽略的。然而有一个特殊情况，当 plot 命令只作用于单个复变量 z 时，则实际绘出实部对应于虚部的关系图形（复平面上的一个点）。这时 plot(z)等价于 plot(real(z), image(z))，其中 z 为矩阵中的一个复向量。

3. 图形的修改及文本标注

MATLAB 中对于同一图形中的多条线，不仅可分别定义其线型，而且可分别选择其色彩，其曲线绘制命令的调用格式如下：

```
plot(x1,y1,选项 1, x2, y2,选项 2,…, xn, yn, 选项 n)
```

其中，x1, x2,…, xn 为 x 轴变量；y1, y2,…, yn 为 y 轴变量，选项见表 2-1。

表 2-1 线型、颜色和标记符号选项

线型		颜色		标识符号			
-	实线	b	蓝色	.	点	s	方块符(Square)
:	虚线	g	绿色	o	圆圈	d	菱形符(Diamond)
-.	点画线	r	红色	x	叉号	v	朝下三角符号
--	双画线	c	青色	+	加号	^	朝上三角符号
		m	品红色	*	星号	<	朝左三角符号
		y	黄色			>	朝右三角符号
		k	黑色			p	六角星符(Pentagram)
		w	白色			h	六角星符(Hexgram)

另外，表中的线型和色彩选项可以同时使用。

例如：

```
>> x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
```

```
'MarkerSize',10)
```

运行程序，输出效果如图 2-10 所示。

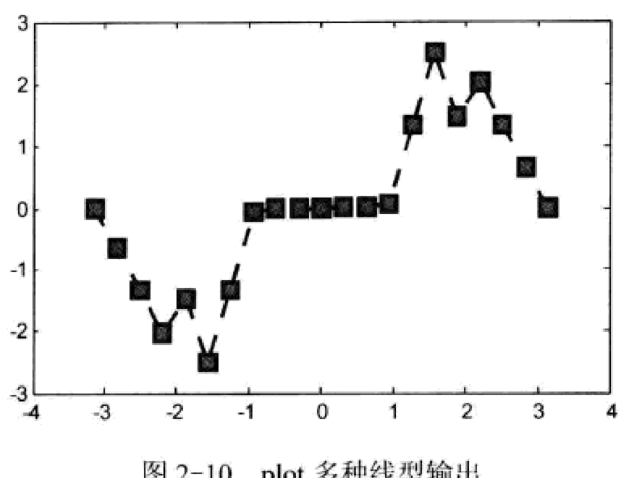


图 2-10　plot 多种线型输出

绘制完曲线后，MATLAB 还提供的特殊绘图函数对屏幕上已有的图形加注释、题头或坐标网格。

例如：

```
>> title('plot example');      %图形标题
>> xlabel('This is x axis');   %x 轴的标注
>> ylabel('This is y axis');   %y 轴的标注
>> grid on                     %增加网格
```

输出带右标注的曲线，效果如图 2-11 所示。

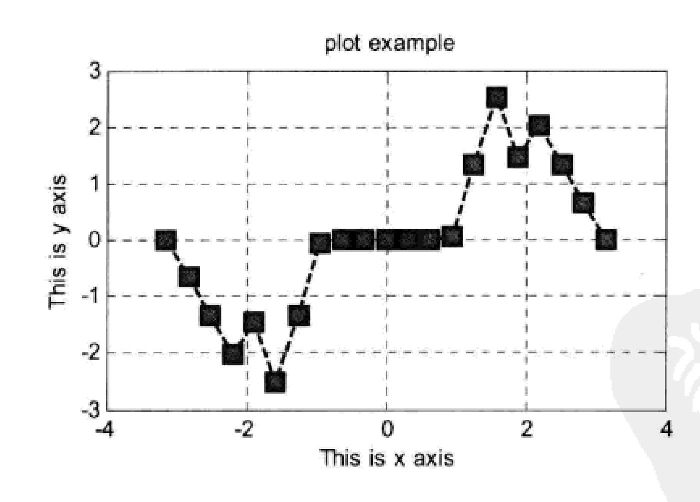


图 2-11　带有标注的 plot 输出曲线

4．图形控制

MATLAB 允许将一个图形窗口分割成 $n \times m$ 部分，对每一部分可以用不同的坐标系统单独绘制图形，窗口分割命令的调用格式如下：

```
subplot(m,n,p)
```

其中，n、m 分别表示将这个图形窗口分割的行列数：p 表示每一部分的代号。例如，想将窗



口分割成 4×3 个部分，则左上角代号为 1，右下角的代号为 12，MATLAB 最多允许 9×9 个窗口的分割。

MATLAB 可以自动根据绘制曲线数范围选择合适的坐标系范围，使得曲线能够尽可能清晰地显示出来。如果觉得自动选择的坐标还不合适时，还可以用手动的方式来选择新的坐标系。其调用格式如下：

```
axis([xmin, xmax, ymin, ymax])
```

另外，MATLAB 还提供了清除图形窗口命令 clg，保持当前窗口的图形命令 hold，放大和缩小窗口命令 zoom 等。

5. 特殊坐标图形

除了基本的绘图命令 plot 外，MATLAB 还允许绘制极坐标曲线，对数坐标曲线，条形图和阶梯图等功能。

极坐标曲线绘制函数的调用格式如下：

```
polar(theta, rho,选项)
```

式中，theta 和 rho 分别为长度相同的角度向量和幅值向量，选项的内容和 plot 函数基本一致。

对数和半对数曲线绘制函数的调用格式分别如下：

```
semilogx(x, y, 选项)    %绘制 x 轴为对数标度的图形
semilogy(x, y, 选项)    %绘制 y 轴为对数标度的图形
loglog(x, y, 选项)      %绘制两个轴均匀对数标度的图形
```

semilogx 仅对横坐标进行对数变换，而纵坐标仍保持线性坐标，而 semilogy 只对纵坐标进行对数变换，而横坐标仍保持线性坐标；loglog 则分别对横纵坐标都进行对数变换（最终得出全对数坐标的曲线来）。选项的定义与 plot 函数完全一致。

【例 2-21】 特殊曲线绘制示例。

```
>> clear all;
x=-1:0.1:1;
subplot(2,2,1);
polar(x,exp(x));
subplot(2,2,2);
semilogx(x,exp(x));
subplot(2,2,3);
semilogy(x,exp(x));
subplot(2,2,4);
loglog(x,exp(x))
```

运行程序，效果如图 2-12 所示。

与线性坐标向量的选取不同，MATLAB 还提供了一个实用的 logspace 函数，按对数等间距的分布来产生一个向量，该函数的调用格式为：

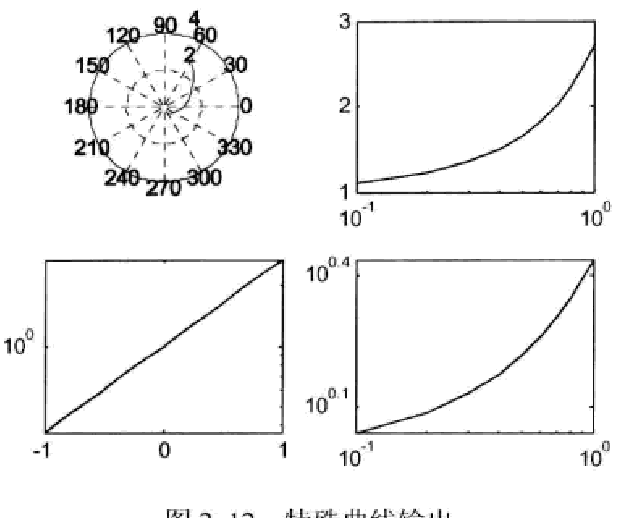




图 2-12　特殊曲线输出

```
x=logspace(n, m, z)
```

式中，10 的 n 次方和 10 的 m 次方，分别表示向量的起点和终点，而 z 表示需要产生向量点的个数，当这个参数忽略时，z 将采用默认值 50。

条形图的绘制函数 bar，其调用格式如下：

```
bar(x, y, 选项)
```

其调用方式是直接绘制由 x 向量和 y 向量给定的条形图，其使用方式与 plot 函数是类似的。

【例 2-22】 条形图绘制函数示例。

```
>> Y = round(rand(5,3)*10);
subplot(2,2,1)
bar(Y,'group')
title 'Group'
subplot(2,2,2)
bar(Y,'stack')
title 'Stack'
subplot(2,2,3)
barh(Y,'stack')
title 'Stack'
subplot(2,2,4)
bar(Y,1.5)
title 'Width = 1.5'
```

运行程序，输出效果如图 2-13 所示。阶梯图的调用 stairs 命令与 bar 命令相类似，唯一的区别在于它输出的图形中没有条形图中所给出的垂直直线，而产生阶梯状图形，这种图形对于统计或绘制数据采集的图十分直观。

6. 利用鼠标绘制图形

MATLAB 允许利用鼠标来单击屏幕，其调用格式如下：

```
[x, y, button]=ginput(n)
```

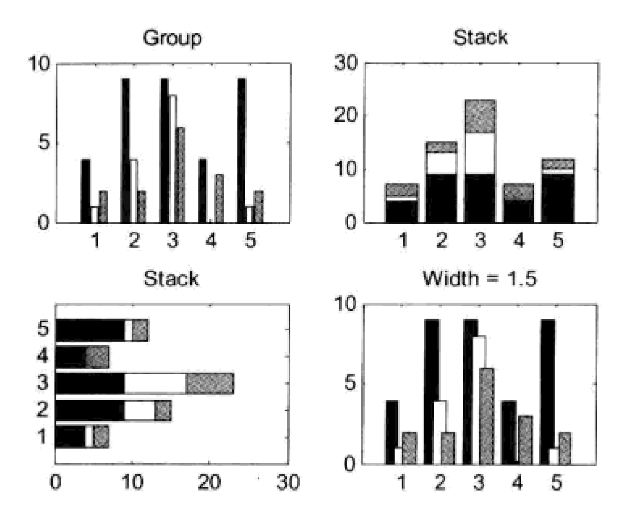


图 2-13　bar 输出效果

其中，n 为选择点的数目；返回的 x、y 向量分别存储被选择上的 n 个点的坐标；button 变为一个 n 向量，它的各个分量为鼠标键的标号，如 button(i)=1，则说明第 i 次按下的是鼠标左键，而该值为 2 或 3 则分别对应于鼠标中键和鼠标右键。

【例 2-23】 用鼠标左键绘制折线，利用鼠标中键或右键中止绘制。

```
>> clg                %清除图形窗口
axis([0 10 0 5]);     %定义坐标轴范围
x1=[];y1=[];
for i=1:99
    [x,y,button]=ginput(1)
    text(x,y,'r.');
    x1=[x1,x];
    y1=[y1,y];
    line(x1,y1);
    if(button~=1);
        break;
    end
end
```

运行程序，用鼠标左键在窗口单击想要的折线，如图 2-14 所示。

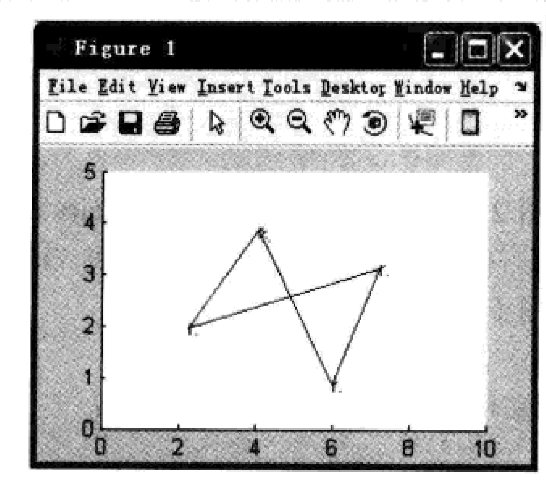


图 2-14　鼠标左键绘制的折线

2.4.2 三维图形绘制

与二维图形相对应，MATLAB 提供了 plot3 函数，它能够在一个三维空间内绘制出三维的曲线，该函数的调用格式如下：

```
plot3(x, y, z, 选项)
```

式中，x、y、z 为维数相同的向量，分别存储曲线的 3 个坐标的值，选项的意义同 plot 函数。

【例 2-24】 利用 plot3 函数绘制三维曲线。

```
>> clear all;
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
grid on
axis square
```

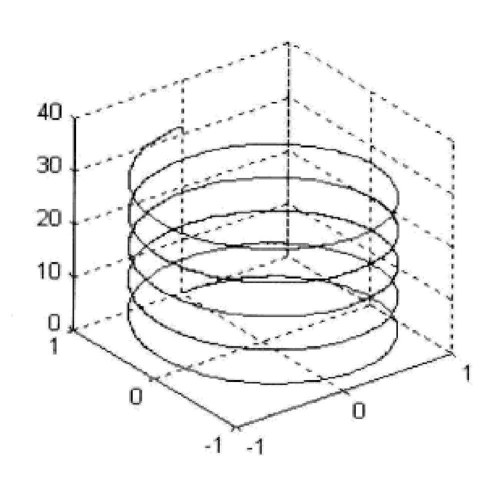


图 2-15 plot3 的三维输出

运行程序，输出效果如图 2-15 所示。

为了使三维图形更漂亮，MATLAB 提供了绘制三维表面网格图的函数。此函数的调用格式如下：

```
mesh(x, y, z, c)
```

式中，x、y、z 分别构成该曲面的 x、y 和 z 向量；c 为色彩矩阵，表示在不同的高度下的色彩范围。如果省略此选项，则会自动地假定 c=z，亦色彩的设定是正比于图形的高度的，这样就可以得出层次分明的三维图形。

meshgrid 函数作为平面网格坐标矩阵的生成。其调用格式如下：

```
[X,Y] = meshgrid(x,y)
[X,Y] = meshgrid(x)
[X,Y,Z] = meshgrid(x,y,z)
```

【例 2-25】 利用 meshgrid 函数创建矩阵。

```
>> [X,Y] = meshgrid(1:3,10:14)
X =
      1      2      3
      1      2      3
      1      2      3
      1      2      3
      1      2      3
Y =
     10     10     10
     11     11     11
     12     12     12
     13     13     13
     14     14     14
```





【例 2-26】 利用 meshgrid 函数创建矩阵生成表面曲面图。

```
>> [X,Y] = meshgrid(-2:.2:2, -2:.2:2);
Z = X .* exp(-X.^2 - Y.^2);
surf(X,Y,Z)
```

运行程序，输出效果如图 2-16 所示。

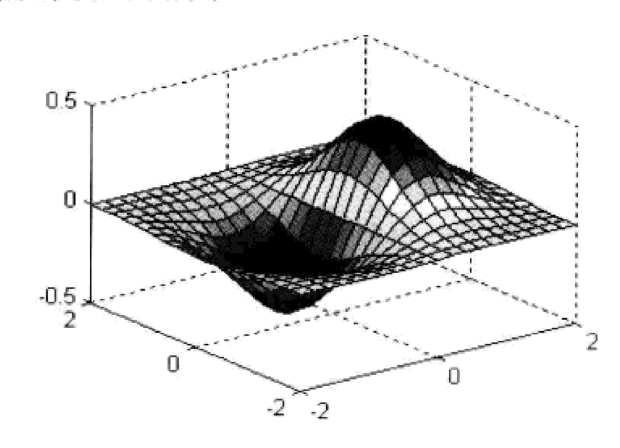


图 2-16 三维曲面

关于三维图形的绘制，常用的其他命令如下：

```
surf(x, y, z)        %绘制三维表面图形
surfc(x, y, z)       %绘制带有等高线的三维表面图形
surfl(x, y, z)       %绘制带有阴影的三维表面图形
coutour(x, y, z)     %等高线图形
```

【例 2-27】 绘制 $z=-\sqrt{x^2+y^2}$ 的网线图和曲面。

其实现的程序代码如下：

```
>> clear all;
x=-9:0.5:9;
y=x;
[x,y]=meshgrid(x,y);
z=-sqrt(x.^2+y.^2);
z=-z.*z;
surf(x,y,z);   %三维曲面图，如图 2-17 所示
pause;
mesh(x,y,z)   %三维曲面图，如图 2-18 所示
```

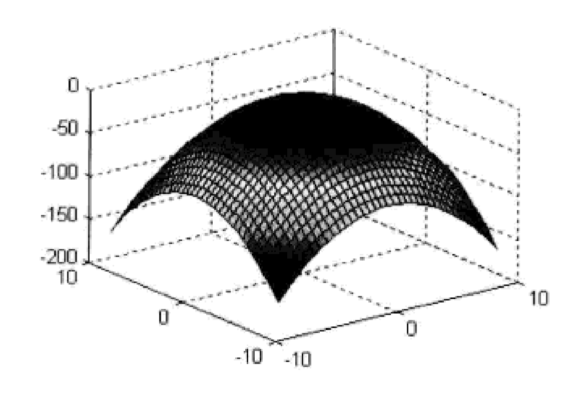


图 2-17 三维曲面

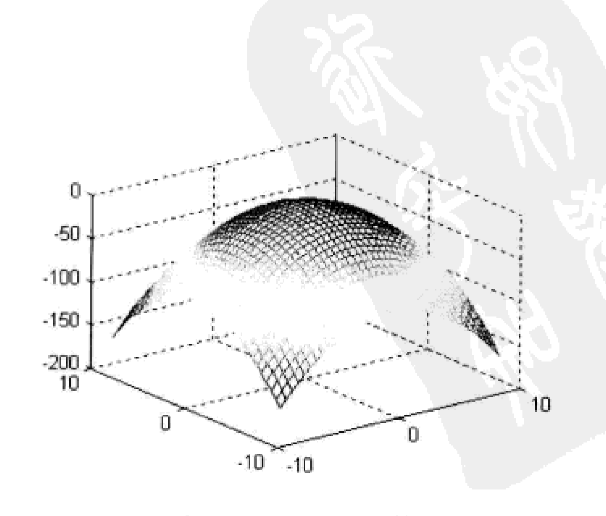
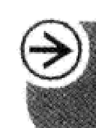


图 2-18 三维网线

【例 2-28】 利用 meshz 绘制三维曲线。

```
>> clear all;
x=-3:.125:3
[X,Y] = meshgrid(x);
Z = peaks(X,Y);
meshz(X,Y,Z)
```

运行程序，输出效果如图 2-19 所示。



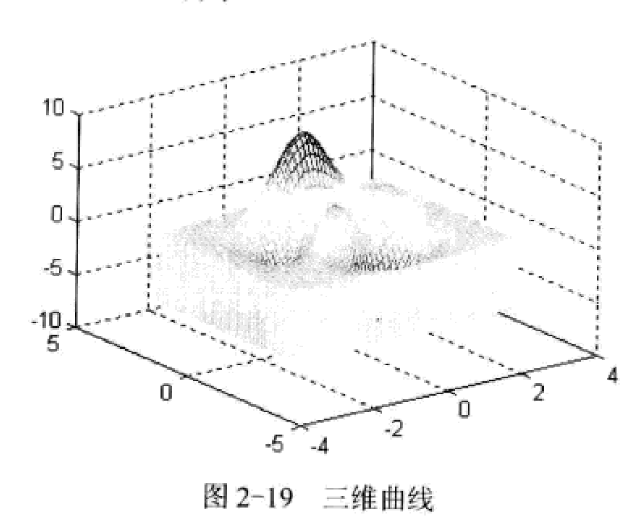


图 2-19 三维曲线

2.5 图形的灯光设置

灯光设置是 MATLAB 中为增加图形场景的视觉效果而提供的一种处理技术。该项技术是通过模拟对象实体在自然光下的明暗特征来实现的。在 MATLAB 中，如果希望产生灯光效果，需要创建一个称为“灯光”（Light）的图形对象。

2.5.1 图形灯光设置对象

利用 MATLAB 中的 light 函数，可以创建灯光对象。每个灯光都包含 3 个非常重要的属性：

- Color：该属性用于指定灯光对象的灯光颜色。
- Style：该属性用于指定光源类型，无限远（默认值）或本地。
- Position：该属性用于指定光线方向（对于无限远光源）或光源位置（对于本地光源）。

其中，Color 属性用于指定来自于光源的具有方向的灯光颜色。所以，图形场景中对象的颜色取决于该对象本身的颜色和光源的颜色。

而 Style 属性用于指定光源是点光源（只要将 Style 值设置为 local 即可。此时灯光对象的光线将以光源所在点为中心向四周辐射），还是无限远光源（只要将 Style 值设置为 infinite 即可。此时该灯光对象的所有光线将是平行的）。

Position 属性用于指定光源在坐标系中的位置。如果光源在无限远处，则该属性就用于指定光源的方向。

灯光对象影响与其在相同坐标下的表面图形对象和片块（Patch）对象。这些对象具有

在有灯光对象时能够改变其自身视觉效果相关属性。

2.5.2 添加灯光效果

在没有光源设置的情况下，MATLAB 绘制图形时自动从颜色方案中对图形进行颜色的配置。例如，利用 membrane 函数绘制的表面图形，其实现代码如下：

```
>> membrane
```

运行程序，输出效果如图 2-20 所示。

添加灯光相关的程序代码如下：

```
>> light('Position',[0 -2 1])
```

该光源为无限远光源，其中矢量[0 –2 1]定义了光源方向，即该光源的方向是顺序通过坐标原点（0, 0, 0）和点（0, –2, 1）。输出效果如图 2-21 所示。

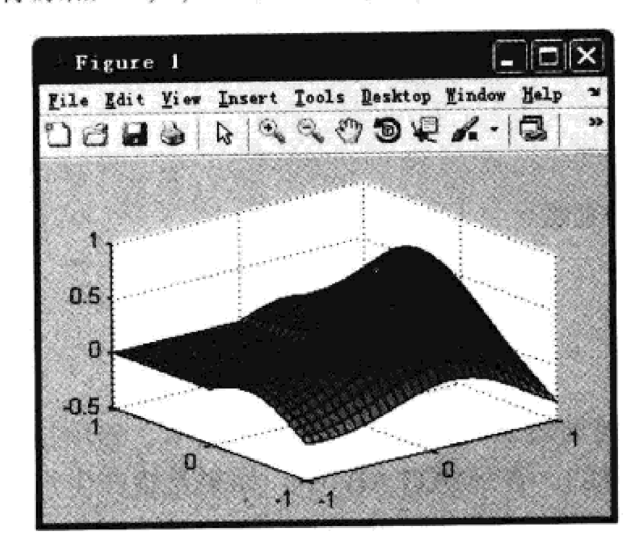


图 2-20　没有灯光的表面图形

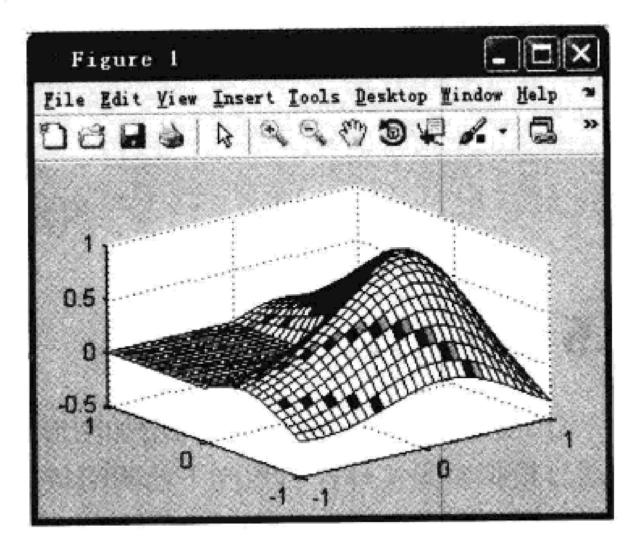


图 2-21　添加了灯光效果后的表面图形

在 MATLAB 中，灯光设置可以增强数学函数的显示效果。例如，利用 ezsurf 命令来计算 -6π 到 6π 之间的 z 函数的值：

$$z = \frac{\sin(\sqrt{x^2 + y^2})}{\sqrt{x^2 + y^2}}$$

代码如下：

```
ezsurf('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)',[-6*pi,6*pi])
```

运行程序，输出效果如图 2-22 所示。

这样绘制的 z 函数没有灯光效果。下面利用 MATLAB 的 lightangle 函数来为该 z 函数的表面图形添加灯光效果，代码如下：

```
>> ezsurf('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)',[-6*pi,6*pi])
view(0,75)
shading interp
```

```
lightangle(-45,30)
set(findobj(gca,'type','surface'),'FaceLighting','phong','AmbientStrength',.3,'DiffuseStrength',.8,'SpecularS
trength',.9,'SpecularExponent',25,'BackFaceLighting','unlit')
```

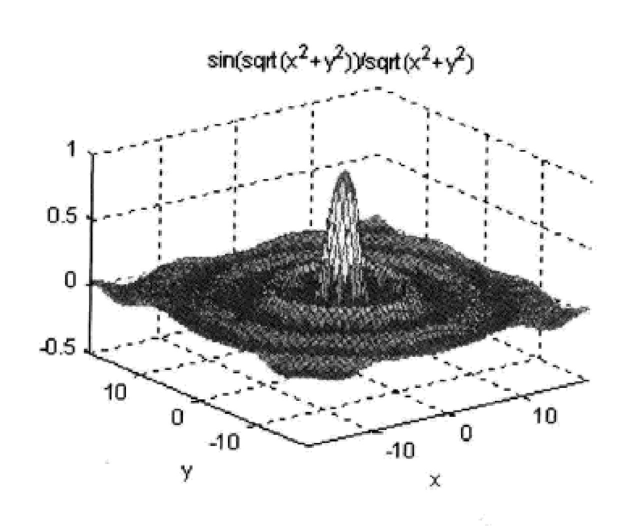


图 2-22 没有灯光效果的 z 函数表面图形

上面的代码中，首先调用 findobj 函数获取表面对象的句柄，然后在此基础上，设置可以受灯光对象影响的属性。

运行程序，输出效果如图 2-23 所示。

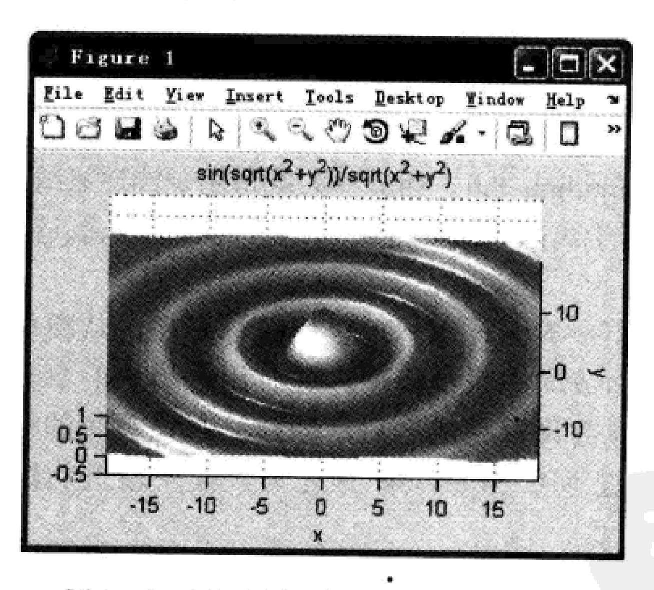


图 2-23 添加灯光效果的 z 函数表面图形



第 3 章　Simulink 仿真基础

Simulink 于 20 世纪 90 年代初由 MathWorks 公司开发，是 MATLAB 环境下对动态系统进行建模、仿真和分析的一个软件包。从字面上看，"Simulink"一词有两层含义，"Simu"表明它可用于系统仿真；"link"表明它能进行系统连接。在该软件环境下，用户可以在屏幕上调用现成的模块，并将它们适当连接起来构成系统的模型，即所谓的可视化建模。建模完成以后，以该模型为对象运行 Simulink 中的仿真程序，可以对模型进行仿真，并可以随时观察仿真结果和干预仿真过程。Simulink 由于功能强大、使用简单方便，已成为应用最广泛的动态系统仿真软件。

3.1　Simulink 概述

3.1.1　Simulink 简介

Simulink 是 MATLAB 的重要组成部分，较为流行的版本有：与 MATLAB 5.2 配用的 Simulink 2.2、与 MATLAB 5.3 配用的 Simulink 3.0、与 MATLAB 6.0 配用的 Simulink 4.0、MATLAB 6.5 配用的 Simulink 5.0，以及与 MATLAB 7.0 配用的 Simulink 6.0、与 MATLAB R2009a 配用 Simulink 8.0。Simulink 既适用于线性系统，也适用于非线性系统;既适用于连续系统，也适用于离散系统和连续与离散混合系统;既适用于定常系统，也适用于时变系统。

Simulink 提供图形用户界面，用户可以用鼠标操作，从模块库中调用标准模块，将它们适当地连接起来以构成动态系统模型，并且用各模块的参数对话框为系统中各模块设置参数。当各模块的参数设置完成之后，即建立起该系统的模型。如果对某一模块没有设置参数，那就意味着使用 Simulink 预先为该模块设置的默认参数值作为该模块的参数。

Simulink 模块库内容十分丰富，除包括输入信号源（Sources）模块库、输出接收（Sinks）模块库、连续系统（Continuous）模块库、离散系统（Discrete）模块库、数学运算（Math Operations）模块库等许多标准模块外，用户还可以自定义和创建模块。

系统的模型建立之后，选择仿真参数和数值算法，便可以启动仿真程序对系统进行仿真，这种操作可以用 Simulink 菜单，也可以用 MATLAB 命令实现。菜单方式对于交互式运行特别方便，而命令方式对于运行一批仿真时很有用。

在仿真过程中，用户可以设置不同的输出方式来观察仿真结果。例如，使用 Sinks 模块库中的 Scope 模块或其他显示模块来观察有关信号的变化曲线，也可以将结果存放在 MATLAB 工作空间中，供以后处理和使用。根据仿真结果，用户可以调整系统参数，观察分析仿真结果的变化，从而获得更加理想的仿真结果。

3.1.2 Simulink 的启动与退出

如果在安装 MATLAB 的过程中选择了 Simulink 组件，则在 MATLAB 安装完成后，Simulink 也就安装了。要注意，Simulink 不能独立运行，只能在 MATLAB 环境中运行。



1. Simulink 的启动

在 MATLAB 的命令窗口中输入 Simulink 或单击 MATLAB 主窗口工具栏上的 Simulink 命令按钮，即可启动 Simulink。Simulink 启动后会显示如图 3-1 所示的 Simulink 模块库浏览器（Simulink Library Browser）窗口。该窗口以树状列表的形式列出了各类模块库，单击所需要的模块，列表窗口的上方会显示所选模块的信息，也可以在模块库浏览器窗口中“Find block”按钮右边的输入栏中，直接输入模块名并单击“Find block”按钮进行查询。

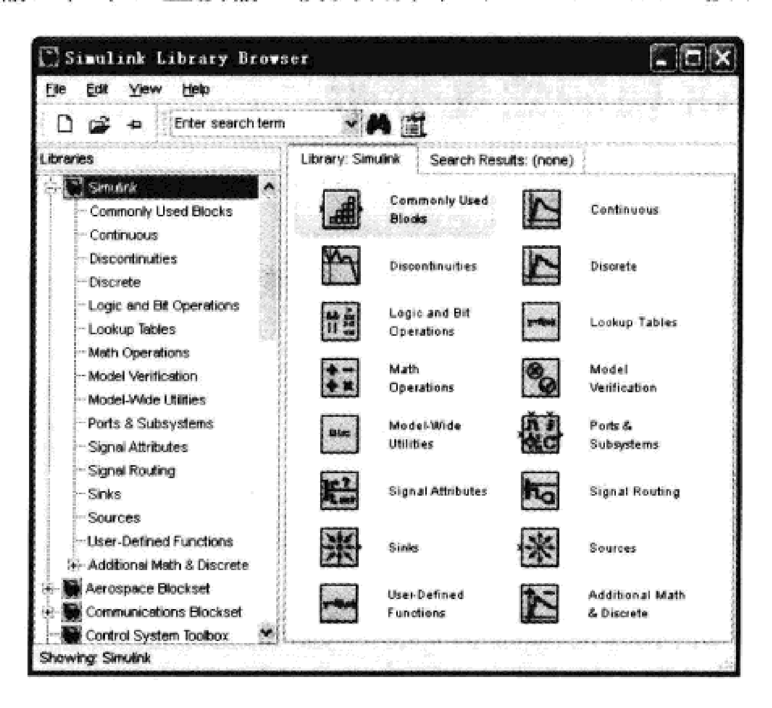


图 3-1　Simulink 模块库浏览器

在 MATLAB 主窗口“File”菜单中，选择“New”子菜单中的“Model”命令，在出现 Simulink 模块库浏览器的同时，还会出现一个名字为“untitled”（模型编辑）窗口，如图 3-2 所示。在启动 Simulink 模块库浏览器后，再单击其工具栏中的“Create a new model”命令按钮，也会弹出“模型编辑”窗口。利用“模型编辑”窗口，可以通过鼠标的拖动操作创建一个模型。

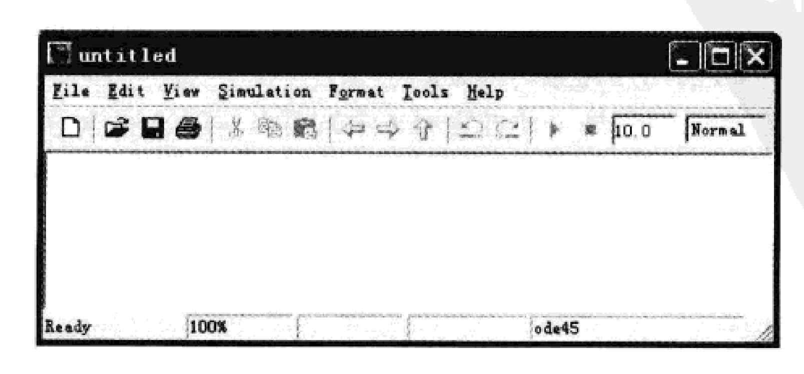


图 3-2　“untitled”（模型编辑）窗口

模型创建完成后，单击模型编辑窗口的“File”菜单项下“Save”或“Save As”命令，可以将模型以模型文件的格式（扩展名为.mdl）存入磁盘。

如果要对一个已经存在的模型文件进行编辑修改，需要打开该模型文件，其方法是：在MATLAB 命令窗口直接输入模型文件名（不要加扩展名.mdl），这要求该文件在当前目录下或在已定义的搜索路径中。单击模块库浏览器窗口或模型编辑窗口的“File”菜单下的“Open”命令，然后选择或输入要编辑模型的名字，也能打开已存在的模型文件。另外，单击模块库浏览器窗口工具栏上的“Open a model”命令按钮或模型编辑窗口工具栏上的“Open model”命令按钮，也能打开已经存在的模型文件。

2. Simulink 的退出

要退出 Simulink，只要关闭所有模型编辑窗口和 Simulink 模块库浏览器窗口即可。

3.2 Simulink 模块处理分析

3.2.1 Simulink 仿真模型构成

一个典型的 Simulink 仿真模型由以下 3 种类型的模块构成。

1. 信号源模块

信号源为系统的输入，它包括常数信号源、函数信号发生器（如正弦波和阶跃函数等）和用户自己在 MATLAB 中创建的自定义信号。

2. 被模拟的系统模块

系统模块作为仿真的中心模块，它是 Simulink 仿真建模所要解决的主要问题。

3. 输出显示模块

系统的输出由显示模块接收。输出显示的形式包括图形显示、示波器显示和输出到文件或 MATLAB 工作空间中 3 种，输出模块主要在 Sinks 模块库中。

构成 Simulink 仿真模型的 3 种模块的关联如图 3-3 所示。

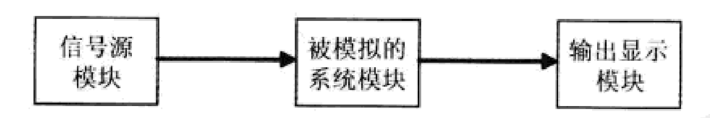


图 3-3　Simulink 仿真模型的结构关联

Simulink 仿真模型的基本特点可归纳如下：

- Simulink 里提供了许多如 Scope（示波器）的接收模块，这里用 Simulink 进行仿真，具有像做实验一般的图形化显示效果。
- Simulink 的模型具有层次性，通过底层子系统可以构建上层母系统。
- Simulink 提供了对子系统进行封装的功能，用户可以自定义子系统的图标和设置参数对话框。

3.2.2 仿真的过程

启动 Simulink 后，便可在 Simulink 中进行建模仿真。Simulink 建模仿真的基本过程如下：

1）打开一个空白的 Simulink 模型窗口。

2）进入 Simulink 模块库浏览界面，将相应模块库中所需的模块拖动到编辑窗口里。具体的操作是：用鼠标左键选中所需要的模块，然后将其拖动到需要创建仿真模型的窗口，松开鼠标，这时所需要的模块就出现在 Simulink 模型窗口中。



3）按照给定的框图修改编辑窗口中模块的参数。在 Simulink 环境下绘制模块，只能绘出带有默认参数的模型，为了满足用户的具体需要，有时还需要对模块参数进行具体设置。

对模块进行参数设置时，首先双击该模块，打开此模块的参数设置对话框。在该参数设置对话框中，既可以查看模块的各项默认参数设置，也可以根据需要修改各项参数设置。

4）将各个模块按给定的框图连接起来，搭建所需要的系统模型。

5）用菜单或命令窗口输入命令进行仿真分析，在仿真的同时，可以观察仿真结果，如果发现有不正确的地方，可以停止仿真，对参数进行修正。

6）如果对结果满意，可以保存模型。

3.3 系统的仿真

3.3.1 模块的基本操作

1. 模块的编辑

（1）添加模块

要把一个模块添加到模型中，首先要在 Simulink 模块库中找到该模块，然后将这个模块拖入模型窗口中即可。

（2）选取模块

要在模型编辑窗口中选择单个模块，只要用鼠标在模块上单击即可，这里模块的角上出现黑色的小方块，拖动这些小方块可以改变模块的大小。要选取多个模块，可以在所有模块所占区域的一角按下鼠标左键不放，拖向该区域的对角，在此过程中会出现虚框。当虚框包住了要选的所有模块后，释放鼠标左键，这时在所有被选模块的角上都会出现小黑块，表示模块都被选中了。

（3）复制与删除模块

在建立系统仿真模型时，可能需要多个相同的模块，这时可采用模块复制的方法。在同一模型编辑窗口中复制模块的方法是：用鼠标单击要复制的模块，按住左键并同时按下〈Ctrl〉键，移动鼠标到适当位置放开鼠标，该模块就被复制到当前位置。

还可以单击模型编辑窗口“Edit”菜单下的“Copy”和“Paste”选项，或单击工具栏上的“Copy”和“Paste”选项按钮来完成复制。

模块复制以后，会发现复制出的模块名称在原名称的基础上又加了编号，这是 Simulink 的约定，每个模型中的模块和名称是一一对应的，每一个模块都有不同的名字。

在不同的模型编辑窗口之间复制模块的方法是，首先打开源模块和目标模块所在的窗口，然后单击要复制的模块，按住左键移动鼠标到相应窗口（不用按住〈Ctrl〉键），然后释放鼠标，该模块就会被复制过来，而且源模块不会被删除。当然，还可以单击模型窗口“Edit”菜单下的“Copy”和“Paste”命令，或单击工具栏上的“Copy”和“Paste”命令来完成复制。

删除模块的方法是：选定模块，按〈Delete〉键或单击“Edit”菜单下的“Cut”选项，或者在模块上单击鼠标右键，在弹出菜单上选择“Cut”命令。注意，用“Cut”命令删除的模块将保存在剪贴板中。

（4）模块外形的调整

1）改变单个模块的大小。首先选中该模块，拖动其周围的 4 个黑色小方块中的任何一个，这时会出现一个虚线的矩形表示新模块的大小，拖到需要的位置后释放鼠标即可。

2）改变整个模型中所有模块的大小。可以单击模型编辑窗口中“View”菜单下的“Zoom In”和“Zoom Out”命令分别用来放大和缩小整个模型；“Fit Selection To View”命令用来将当前选中的模块或当前系统放大到整个窗口大小来观察；“Normal（100%）”命令用来将整个模型恢复到原始的正常大小。

3）调整模块的方向。首先选定模块，然后单击模型编辑窗口“Format”菜单下的“Rotate Block”命令使模块顺时针方向旋转 90°，单击“Flip Block”命令使模块旋转 180°。显然两次旋转 90°与一次旋转 180°的操作效果是一样的。

4）改变模块的颜色。首先选定模块，然后单击“Format”菜单下的“Foreground Color”命令，选择模块的前景色，即模块的图标、边框和模块名的颜色，使模块产生阴影效果。单击“Format”菜单下的“Background Color”命令，选择模块的背景色，即模块的背景填充色。单击“Format”菜单下的“Screen Color”命令，可以用来改变模型的背景色。

5）给模块加阴影。首先选定模块，然后单击“Format”菜单下的“Show Drop Shadow”命令使模块产生阴影效果。

（5）模块名的处理

1）隐藏模块名。首先选定模块，然后单击“Format”菜单下的“Hide Name”命令，模块名就会被隐藏，同时“Hide Name”改为“Show Name”。选择“Show Name”命令就会使模块隐藏的名字显示出来。

2）修改模块名。单击模块名的区域，这时会在此处出现编辑状态的光标，在这种状态下能够对模块名随意修改。模块名和模块图标中的字体也可以更改，方法是选定模块，单击“Format”菜单下的“Font”命令，这时系统弹出“Set Font”对话框，在对话框中选择想要的字体。

模块名的位置有一定的规律，当模块的接口在左右两侧时，模块名只能位于模块的上、下两侧，默认在下侧；当模块的接口在上、下两侧时，模块名只能位于模块的左、右两侧，默认在左侧。因此模块名只能从原位置移动到相对的位置。可以用鼠标拖动模块名到其相对的位置；也可以选定模块，单击“Format”菜单下的“Flip Name”命令实现相对的移动。

2. 模块的连接

当设置了各个模块后，还需要把它们按照一定的顺序连接起来才能组成一个完整的系统模型。

（1）连接两个模块

从一个模块的输出端到另一个模块的输入端，这是 Simulink 仿真模块最基本的连接情况。方法是先移动光标到输出端，光标箭头会变成十字形光标，这时按住鼠标左键，移动鼠标到另一个模块的输入端，当十字形光标出现重影时，释放鼠标左键就完成了连接。

如果两个模块不在同一水平线上，连线是一条折线。若要用斜线表示，则需要在连线

后，选中连线，再按住〈Shift〉键进行拖动。两个模块的连接效果如图 3-4 所示。

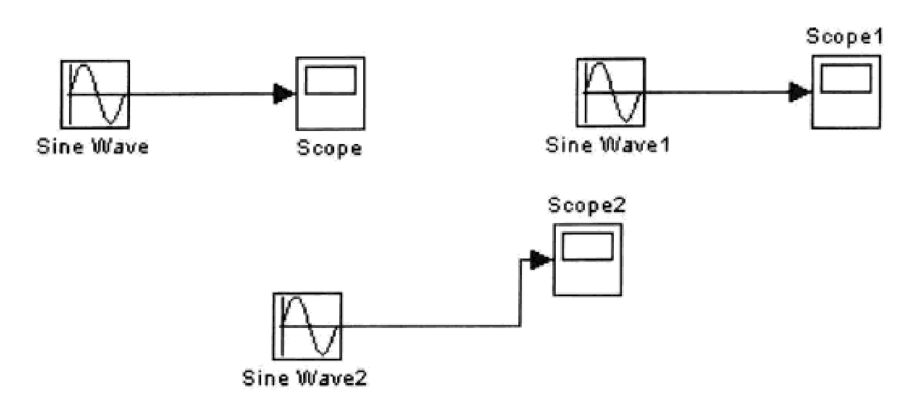


图 3-4　两个模块的连接种类



（2）模块间连线的调整

调整模块间连线位置可采用鼠标拖放操作来实现。先把鼠标移动到需要移动的线段的位置，按住鼠标左键，移动鼠标到目标位置，释放鼠标左键。

还有一种情况，要把一条直线分成斜线段。调整方法和前一种情况类似，不同之处在于按住鼠标左键之前要先按下〈Shift〉键，出现黑色小方块之后，拖动小方块到目标位置后释放鼠标和〈Shift〉键。

（3）标注连线

为了使模型更加直观、可读性更强，可以为传输的信号作标记。

若要给传输的信号作标记，可以双击要加标记的连线，将出现一个小文本编辑框，在里面输入标注文本，这样就建立了一个信号标记。

单击“Format”菜单下的“Port/Signal Displays”子菜单中的一些命令，可以给连线加不同的标志。例如，“Port Data Types”命令在连线上显示前一个模块输出的数据类型；“Signal Dimensions”命令在连线上标出输入/输出信号的维数；“Wide Nonscalar Lines”命令定义模型中的哪些线传递的是向量信号，传输向量的线要粗些。

（4）连线的分支

在仿真过程中，经常需要把一个信号输送到不同的模块，这时就需要从一根连线中分出一根连线。操作方法是：在先连好一条线之后，把鼠标移到分支点的位置，先按下〈Ctrl〉键，然后按住鼠标拖动目标模块的输入端，释放鼠标和〈Ctrl〉键。

（5）删除连线

要删除某条连线，可单击该连线，然后单击“Cut”命令或按〈Delete〉键即可。

3．模块的参数设置

在完成模块的信号线连接并建立起系统的 Simulink 仿真模型后，需要设置模块的参数。在 Simulink 模型里，双击需要修改参数的模块，系统弹出“参数设置”对话框。例如，在图 3-4 中，双击“Sine Wave”模块，系统弹出如图 3-5 所示的“参数设置”对话框。在这里可设置正弦波信号参数，如将正弦信号的默认幅度值 1 设置为 2。

接着设置示波器模块的参数，示波器模块用来显示仿真后的输出结果。双击“Scope”模块，系统弹出“Scope”窗口，如图 3-6 所示。这是一个示波器显示窗口，窗口中标明了 x、y 轴坐标，用户可以根据需要改变坐标轴的显示参数，或者不显示参数。

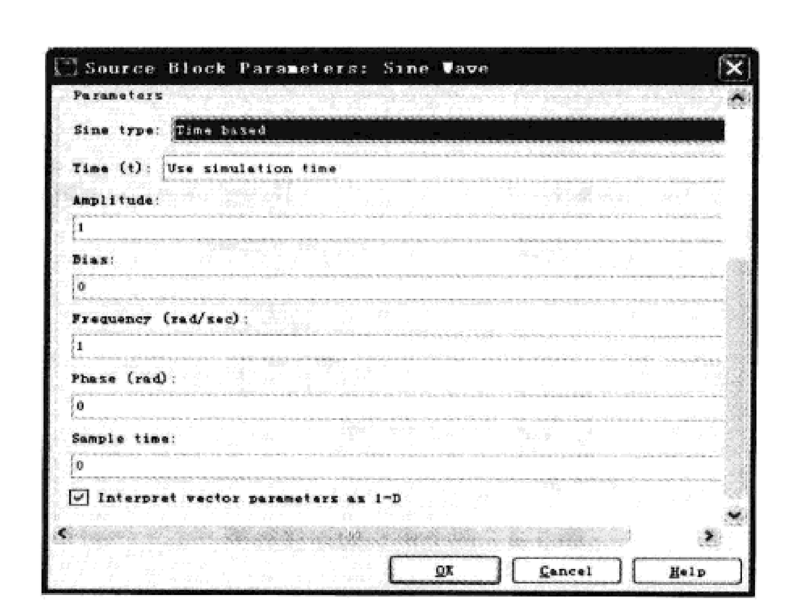


图 3-5　Sine Wave 模块“参数设置”对话框

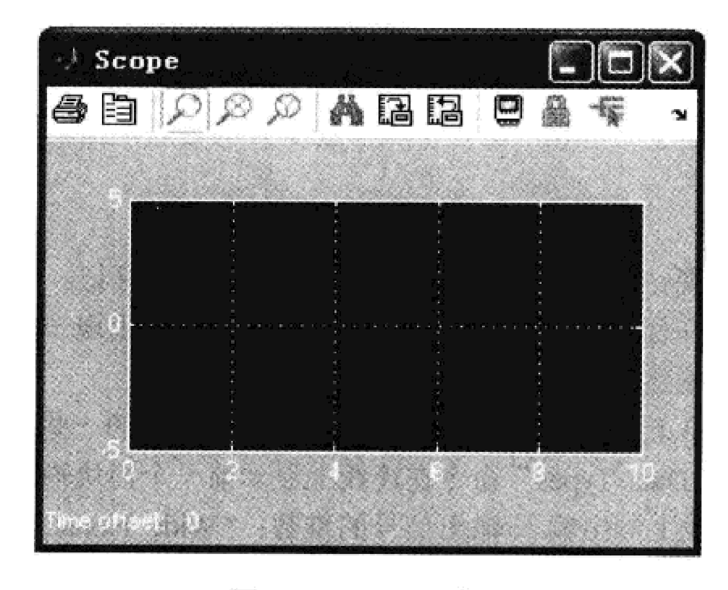


图 3-6　“Scope”窗口

在示波器显示窗口上有一个工具条，工具条上各按钮从左到右的作用为：打印、示波器参数、同时放大 x、y 坐标轴、放大 x 坐标轴，放大 y 坐标轴、自动缩放、保存坐标轴设置、恢复坐标轴设置、浮动示波器、释放坐标轴选项、信号选择器。

示波器模块可以有多个坐标轴（每个输出端口对应一个 x、y 坐标轴），所有的坐标轴都有相同的时间范围，但可对应不同的 y 轴坐标，用户可以调整示波器显示的时间量值及输出值范围，也可以移动或改变示波器窗口的大小。此外，在 Simulink 进行仿真的过程中，用户也可以更改示波器的参数值。

在示波器窗口单击鼠标右键，系统弹出一个菜单，选择“Axes Properties”命令，打开“Scope properties”（示波器属性）对话框，如图 3-7 所示。用户可以在对话框中的“Y-min”和“Y-max”文本框中输入 y 坐标轴的最小值与最大值。此外，“Title”文本框要求输入示波器显示波形的名称，这里也可以输入信号的名称。

接下来，单击示波器工具栏上的图标，打开“Scope properties”（示波器属性）对话框，这个对话框有两个选项卡，即 General 和 Data history，分别如图 3-8a 和图 3-8b 所示。

在“General”选项卡上可以设置坐标轴参数、时间范围和坐标轴标记，也可以选择浮动示波器。

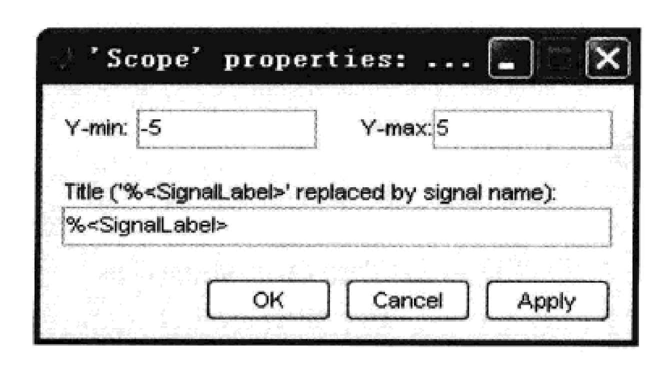
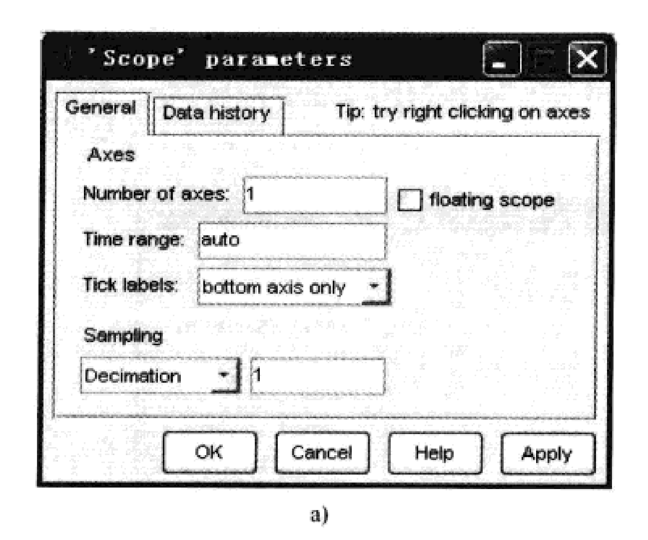
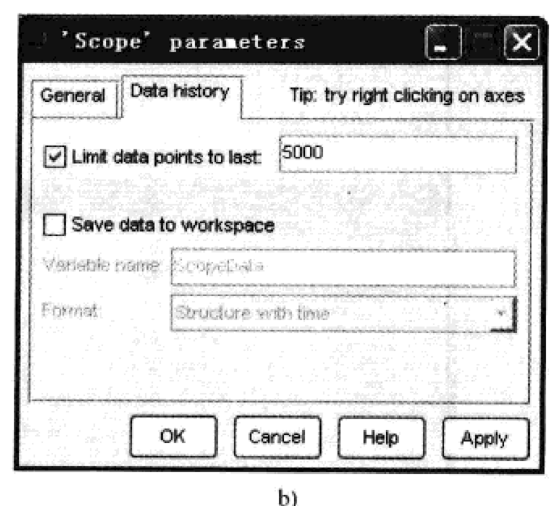


图 3-7 “Scope properties”对话框





a)



b)

图 3-8 “Scope properties”对话框

a) General 选项卡 b) Data history 选项卡

下面介绍各文本框内参数的设置方法。

1）“Number of axes”文本框：设置 y 坐标轴的数目，除了浮点示波器之外，示波器模块对坐标轴的数目没有限制，所有的 y 坐标轴都共有相同的时间轴（即 x 轴），但 y 坐标轴都是独立的。需要说明的是，坐标轴的数目等于输入端口的数目。

2）“Time range”文本框：设置 x 轴的时间范围，单击为 s，它表示示波器窗口只显示指定时间范围内的波形，该参数的默认值为 auto，它表示设置 x 轴的时间范围为仿真持续时间，该文本框不允许输入变量名。本示例使用默认设置。

3）“Tick labels”下拉列表框：它表示是否对坐标轴标上标记，有 3 个选项，即 all（标记所有的坐标轴）、none（所有的坐标轴均不标记）和 bottom axis only（只标记底部的 x 轴）。本示例中选择“all”选项。

4）“floating scope”复选框：选中这个复选框将会把示波器模块变为浮动示波器，浮动示波器是可以显示一条或多条曲线的示波器，在 Simulink 的 Sinks 子模块库中有 Floating Scope 模块，可以直接拖动该模块建立浮动示波器。可以单击示波器窗口上的图标来选择浮动示波器上需要显示的信号曲线。

5）“Sampling”下拉列表框：如果选择“Decimation”选项，则在右侧的文本框内输入一个数值来指定小数部分。若要显示采样间隔内的数据，选择“Sample time”选项，并输入数值。

6）“Limit data points to last”复选框：该选项可以限制保存到工作空间的数据点的个数，示波器依据这个选项执行放大和自动缩放操作，如果数据点的个数限制为1000，而仿真过程产生了2000个数据点，那么示波器只会依据后1000个数据点来绘制显示波形。

3.3.2 仿真参数的设置

在系统仿真过程中，事先必须对仿真算法、输出模式等各种仿真参数进行设置。其方法是：打开系统仿真模型，单击模型编辑窗口“Simulation”菜单下的“Configuration Parameters”命令，打开仿真参数对话框。仿真参数设置是Simulink仿真的一优点、亮点，也是一个难点。Simulink默认的仿真参数配置是，起始时间“Start time”为0.0s，终止时间“Stop time”为10.0s。求解器设置如下：最大步长、最小步长、初始步长由系统自动设定，仿真算法为“ode45”（四/五阶的龙格-库塔法，适用于连续系统的仿真），相对误差限为0.001，绝对误差限由系统自动设定。对于图3-9所示系统框图Simulink仿真，仿真参数对话框中可以设置9个选项：

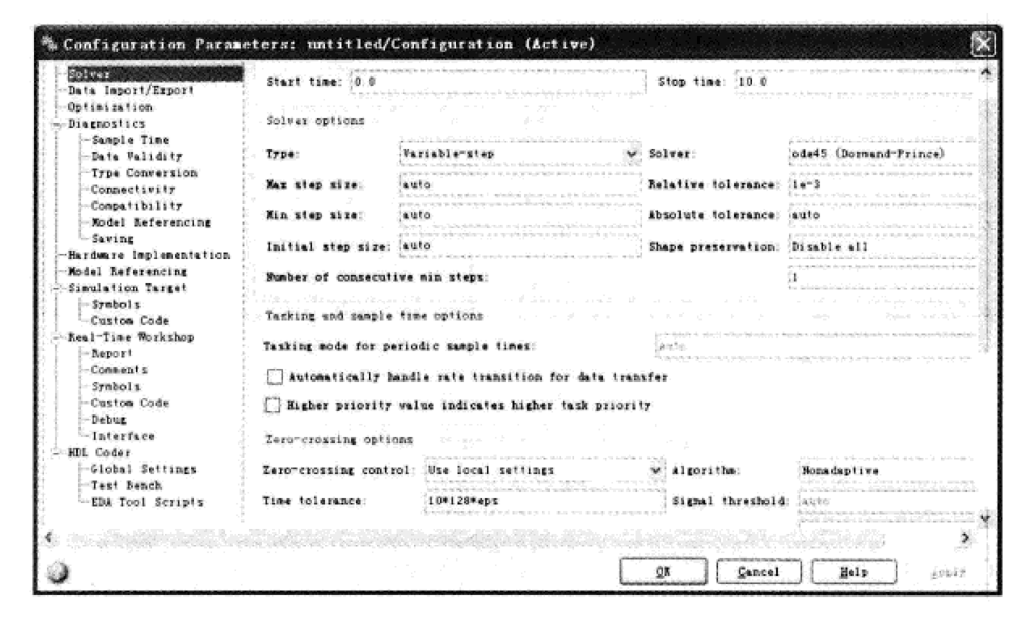


图3-9　仿真参数设置对话框

1）Solver：用于设置仿真起始时间和停止时间，选择微分方程求解算法并为其规定参数。

2）Data Import/Export：用于设置Simulink与MATLAB工作空间交换数据的有关选项。

3）Optimization：用于设置仿真的优化参数。

4）Diagnostics：用于设置在仿真过程中出现各类错误时发出警告的等级。

5）Hardware Implementation：用于设置仿真硬件特性。

6）Model Referencing：用于设置模型引用的有关参数。

7）Simulation Target：用于确定模型的仿真目标。

8）Real-Time Workshop：用于设置若干实时工具中的参数。如果没有安装实时工具箱，则将不出现该选项。

9）HDL Coder：用于生成产品代码的工作。

下面对“Solver”选项和“Data Import/Export”选项进行介绍。



1．Solver 选项

（1）仿真时间（Simulation Time）设置

这里所指的时间概念与真实的时间并不一样，只是计算机仿真中对时间的一种表示，比如 10s 的仿真时间，如果采样步长定为 0.2，则需要执行 50 步，若把步长减小，则采样点增加，那么实际的执行时间就会增加。

需要设置的有仿真开始时间（Start Time）和仿真结束时间（Stop Time）。一般仿真开始时间设为 0，而结束时间则视不同的情况进行而定。

总体来说，执行一次仿真要耗费的时间取决于很多因素，包括模型的复杂程度、解法器及其步长的选择、计算机时钟的速度等。

（2）仿真步长模式设置

用户在“Type”下拉选项框中指定仿真的步长选择方式，可供选择的有“Variable-step”（变步长）和“Fixed-step”（固定步长）方式。

选择变步长模式则可以在仿真过程中改变步长，提供误差控制和过零检测选择。固定步长模式则可以在仿真过程中提供固定的步长，不提供误差控制和过零检测。

（3）解法器设置

用户在“Solver”下拉列表框中可以选择变步长模式解法器或固定步长模式解法器。变步长模式解法器有：Discrete、ode45、ode23、ode113、ode15s、ode23s、ode23t 和 ode23tb。下面简要概述一下这些解法器的含义。

1）Discrete：当 Simulink 检查到模型没有连接状态时使用它。

2）ode45：默认值，表示四/五阶龙格-库塔法，适用于大多数连续或离散系统，但不适用于刚性（Stiff）系统。它是单步解法器，即在计算 $y(t_n)$ 时，它仅需要最近处理时刻的结果 $y(t_{n-1})$。一般来说，面对一个仿真问题时最好首先试试 ode45。

3）ode23：表示二/三阶龙格-库塔法，它在误差限要求不高和求解的问题不太难的情况下，可能会比 ode45 更有效。它也是一个单步解法器。

4）ode113：表示一种阶数可变的解法器，它在误差允许的情况下通常比 ode45 有效。ode113 是一种多步解法器，即在计算当前时刻输出时，它需要以前多个时刻的解。

5）ode15s：表示一种基于数字微分公式的解法器（NDFs），它也是一种多步解法器。适用于刚性系统，当用户估计要解决的问题比较困难、不能使用 ode45 时，或者即使使用 ode45 效果也不好时，就可以用 ode15s。

6）ode23s：表示一种单步解法器，专门应用于刚性系统，在弱误差允许下的效果优于 ode15s。它能解决某些 ode15s 所不能有效解决的刚性问题。

7）ode23t：表示梯形规则的一种自由插值实现。这种解法器适用于求解适度刚性问题而用户又需要一个无数字振荡的解法器的情况。

8）ode23tb：表示 TR-BDF2 的一种实现，TR-BDF2 是具有两个阶段的隐式龙格-库塔公式。

固定步长模式解法器有：Discrete、ode5、ode4、ode3、ode2、ode1 和 ode14x。

1）Discrete：表示一种实现积分的固定步长解法器，它适合于离散无连续状态的系统。

2）ode1：表示欧拉法。

3）ode2：表示改进的欧拉法。

4）ode3：表示固定步长的二/三阶龙格-库塔法。

5）ode4：表示四阶龙格-库塔法，具有一定的计算精度。

6）ode5：默认值，是 ode45 的固定步长版本，适用于大多数连续或离散系统，不适用于刚性系统。

7）ode14s：表示固定步长的隐式外推法。

（4）变步长的参数设置

对于变步长模式，用户常用的设置有：最大和最小步长参数、相对误差和绝对误差、初始步长和过零控制。默认情况下，步长自动确定，用 auto 值表示。

1）Max step size（最大步长参数）：决定解法器能够使用的最大时间步长，它的默认值为“仿真时间/50”，即整个仿真过程中至少取 50 个取样点，但这样的取法对于仿真时间较长的系统则可能带来取样点过于稀疏的问题，继而使仿真结果失真。

一般建议对于仿真时间不超过 15s 的采用默认值即可，超过 15s 的每秒至少保证 5 个采样点，对于超过 100s 的，每秒至少保证 3 个采样点。

2）Min step size（最小步长参数）：用来规定变步长仿真时使用的最小步长。

3）Initial step size（初始步长参数）：一般建议使用 auto 默认值。

4）Relative tolerance（相对误差）：指误差相对于状态的值，是一个百分比，默认值为 1e-3，表示状态的计算值要精确到 0.1%。

5）Absolute tolerance（绝对误差）：表示误差值的门限，或者是在状态值为零的情况下可以接受的误差。如果它被设成了 auto，那么 Simulink 为每一个状态设置的初始绝对误差为 1e-6。

6）Shape prvservation（模型的保存）：建议保存为 Disable all。

（5）固定步长的参数设置

对于固定步长模式，用户常用的设置有以下几个。

1）Multiasking：选择这种模式时，当 Simulink 检测到模块间非法的采样速率转换时，系统会给出错误提示。所谓的非法采样速率转换是指两个工作在不同采样速率的模块之间的直接连接。

在实时多任务系统中，如果任务之间存在非法采样速率转换，那么就有可能出现一个模块的输出在另一个模块需要时却无法利用的情况。

通过检查这种转换，Multiasking 将有助于用户建立一个符合现实的多任务系统的有效模型。使用速率转换模块可以减少模型中的非法速率转换。

Simulink 提供了两个这样的模块：unit delay 模块和 zero-order hold 模块。对于从慢速率到快速率的非法转换，可以在慢输出端口和快输入端口插入一个单位延时（unit delay）模块。对于快速率到慢速率的转换，则可以插入一个零阶采样保持器（zero-order hold）。

2）Singletasking：这种模式不检查模块间的速率转换，它在建立单任务系统模型时非常有用，在这种系统中不存在任务同步问题。

3）Auto：选择这种模式时，Simulink 会根据模型中模块的采样速率是否一致，自动决定切换到 Multiasking 模式或 Singletasking 模式。



2. Data Import/Export 选项

Data Import/Export（工作空间数据导入/导出）设置主要在 Simulink 与 MATLAB 工作空间交换数值时进行有关选项设置，可以设置 Simulink 和当前工作空间的数据输入和输出。

通过设置，可以从工作空间输入数据、初始化状态模块，也可以把仿真结果、状态变量、时间数据保存到当前工作空间，它包括“Load from workspace”、“Save to workspace”和“Save options”3 个选择项。

（1）Load from workspace

选中前面的复选框即可从 MATLAB 工作空间获取时间和输入变量，一般时间变量定义为 *t*，输入变量定义为 *u*。“Initial state”用来定义从 MATLAB 工作空间获得的状态初始值的变量名。

Simulink 通过设置模型的输入端口，实现在仿真过程中从工作空间读入数据。常用的输入端口模块为信号与系统模块库（Signals & Systems）中的 In1 模块，设置其参数时，选中 input 前的复选框，并在后面的文本框输入数据的变量名，可以用命令窗口或 M 文件编辑器输入数据。Simulink 根据输入端口参数中设置的采样时间读取输入数据。

（2）Save to workspace

用来设置保存在 MATLAB 工作空间的变量类型和变量名。可以选择保存的选项有：时间、端口输出、状态和最终状态。选中某一选项前面的复选框，并在该选项后面的文本框输入变量名，就会把相应数据保存到指定的变量中。常用的输出模块为信号与系统模块库（Signals & Systems）中的 Out1 模块和输出方式库（Sink）中的 To Workspace 模块。

（3）Save options

用来设置存往工作空间的有关选项。

1）Limit rows to last：用来设定 Simulink 仿真结果最终可存往 MATLAB 工作空间的变量的规律，对于向量而言即其维数，对于矩阵而言即其秩。

2）Decimation：设定一个亚采样因子，它的默认值为 1，也就是对每一个仿真时间点产生值都保存。若将其设置为 2，则每隔一个仿真时刻才保存一个值。

3）Format：用来说明返回数据的格式，包括 Array（数组）、Structure（结构体）及 Structure with time（带时间的结构体）。

4）Signal logging name：用来保存仿真记录的变量名。

5）Output options：用来生成额外的输出信号数据。它有以下 3 个选项：

- Refine output：这个选项可以理解成精细输出，其意义是在仿真输出太稀疏时，Simulink 会产生额外的的精细输出，就像插值处理一样。
- Produce additional output：它允许用户直接指定产生输出的时间点。一旦选择了该选项，则在它的右边出现一个“output times”文本框，用户在这里指定额外的仿真输出点，它既可以是一个时间向量，也可以是表达式。与精细因子相比，这个选项会改变仿真的步长。
- Produce specified output only：它的意思是让 Simulink 只在指定的时间点上产生输出。为此解法器要调整仿真步长以使之与指定的时间点生重合。这个选项在比较不同的仿真时可以确保它们在相同的时间输出。

6）Refine factor：用来指定仿真步长之间产生数据的点数。

用户可以在“Refine factor”中设置仿真时间步间插入的输出点数，产生更光滑的输出曲线，改变精细因子比减小仿真步长更有效。精细输出只能在变步长模式中才能使用，并且在 ode45 效果最好。

3.4 Simulink 模块库简介

3.4.1 常用模块库

常用模块库（Commonly Used Blocks）是为了加快建模速度、节省建模过程中寻找模块的时间而将最常用的基本模块集中放在一起形成的。在初学 Simulink 建模与仿真时，是使用最为频繁的模块库。在 Simulink 浏览器中，单击左侧列表中的“Commonly Used Block”节点，可打开常用模块库，如图 3-10 所示。也可以鼠标右键单击“Commonly Used Block”节点，在弹出的快捷菜单中选择“Open Commonly Used Block library”选项，系统弹出“独立的常用模块库”窗口，如图 3-11 所示。

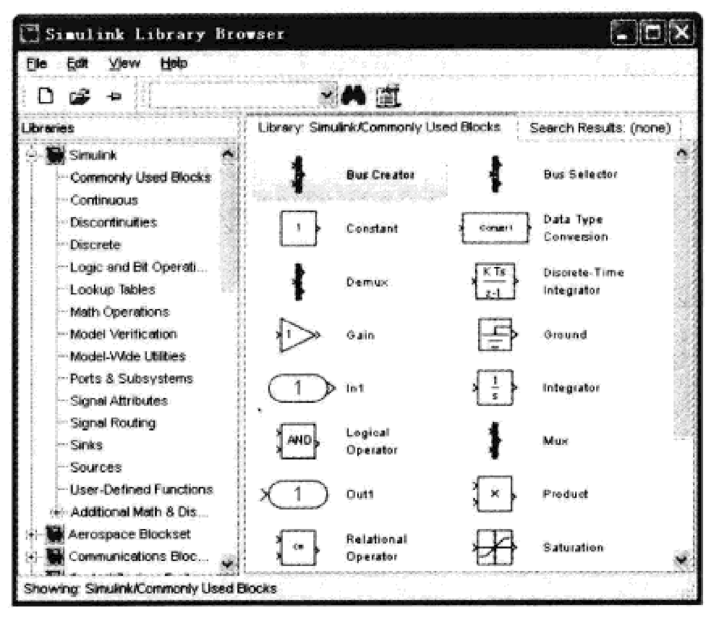


图 3-10 常用模块库

从图 3-10 和图 3-11 中可见，常用模块库包括 Bus Creator（总线信号生成器）模块、Commonly Bus Selector（常数总线信号选择器）模块、Constant（常数模块）、Data Type Conversion（数据类型转换）模块、Demux（信号分离器）模块、Discrete-Time Integrator（离散时间积分）模块、Gain（增益）模块、Ground（信号）模块、In1（输入接口）模块、Integrator（积分）模块、Logic Operator（逻辑操作）模块、Mux（信号合成器）模块、Out1（输出接口）模块、Product（乘法）模块、Subsystem（子系统）模块、Sum（求和）模块、Switch（开关转换）模块、Terminator（信号终端）模块、Unit Delay（单位延迟）模块。

1. 总线信号生成与总线信号选择模块

Bus Creator（总线信号生成器）模块用于将多个信号合成为一个总线信号，常用于子系统接口信号传递；Bus Selector（总线信号选择器）模块用来选择总线信号中的一个或多个，如图 3-12 所示，有 3 种输入信号，分别为正弦信号、阶跃信号和脉冲信号。为便于观察，设阶

跃信号阶跃时间为 1.2s，初始值为 0，终止值为 0.5，其他信号取默认值，效果如图 3-13 所示。双击“Bus Creator”模块，系统弹出“参数设置”对话框，将输入信号数改为 3，效果如图 3-14 所示。双击“Bus Selector”模块，系统弹出“参数设置”对话框，选择输出信号 1 和信号 3，效果如图 3-15 所示。同时双击“Scope”模块，再单击“Parameters”工具，系统弹出“Scope 模块参数设置”对话框，将坐标系数设置为 2。同时设置 Scope1 观察信号合成结果，如图 3-16 所示。运行仿真，示波器输出结果如图 3-17 所示。



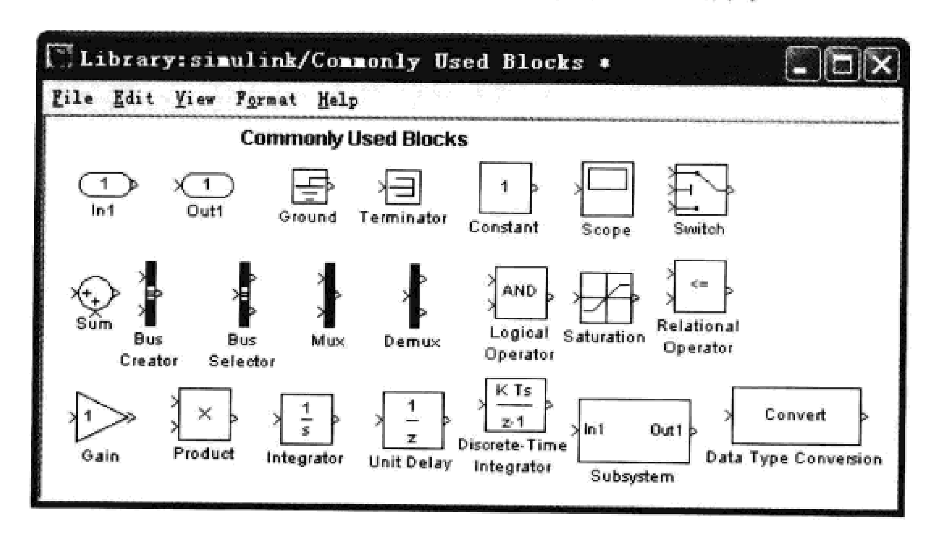


图 3-11 独立的常用模块库

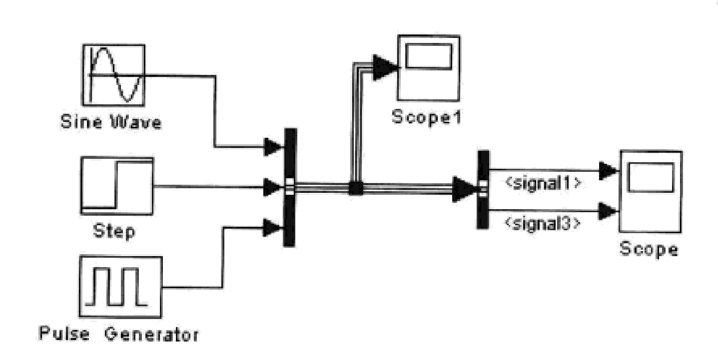


图 3-12 信号合成与选择

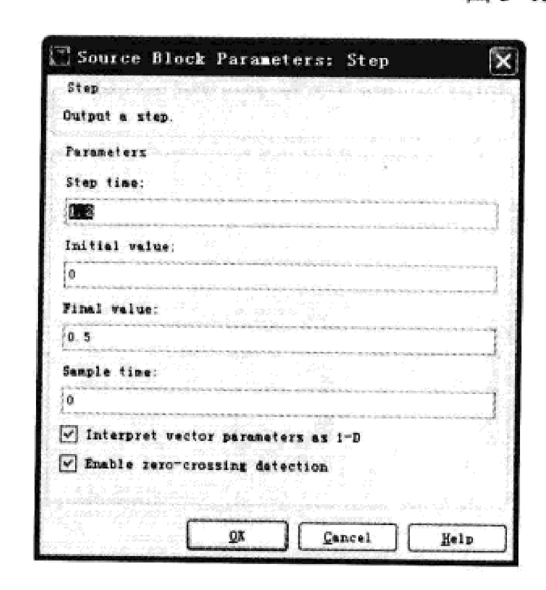


图 3-13 阶跃信号参数设置

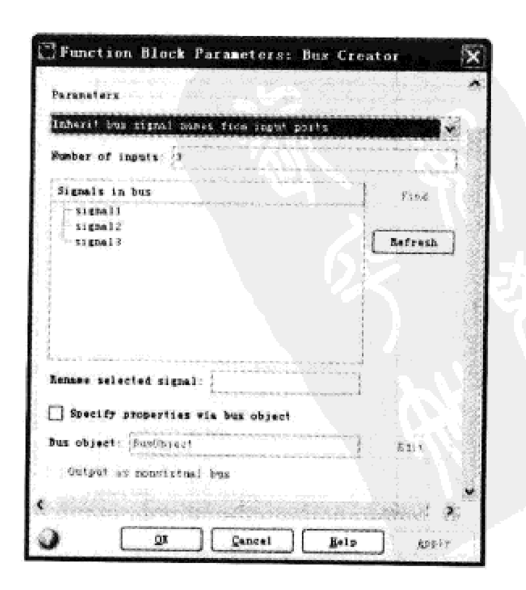


图 3-14 总线信号生成器参数设置

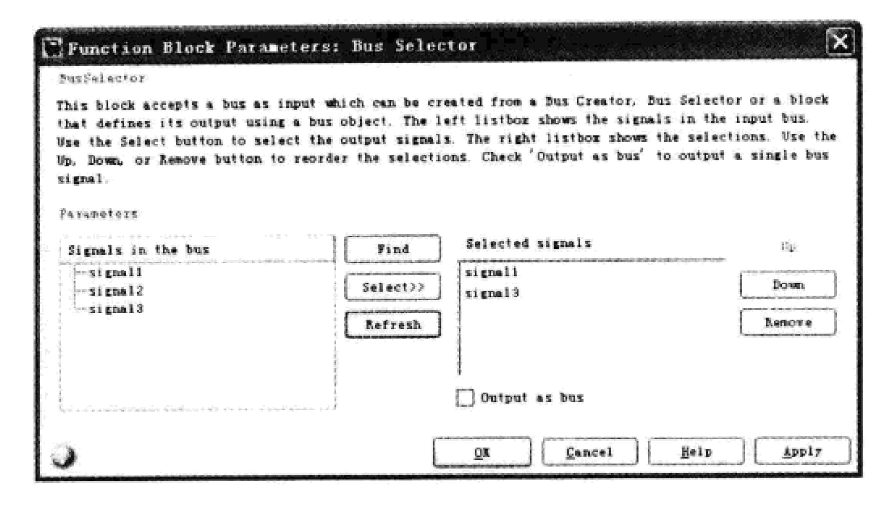


图 3-15　信号选择模块参数对话框

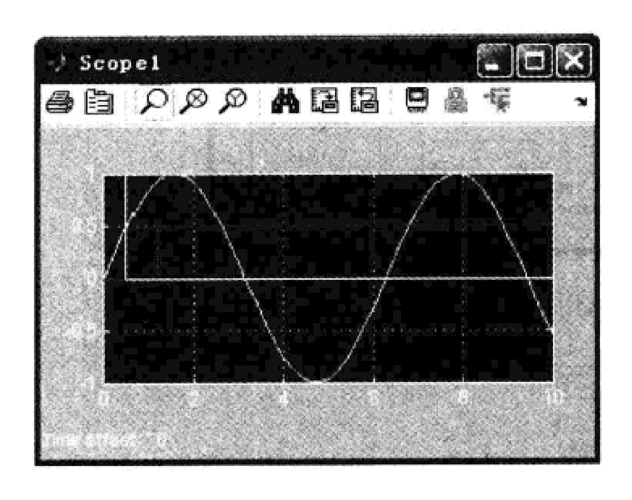


图 3-16　信号合成输出结果

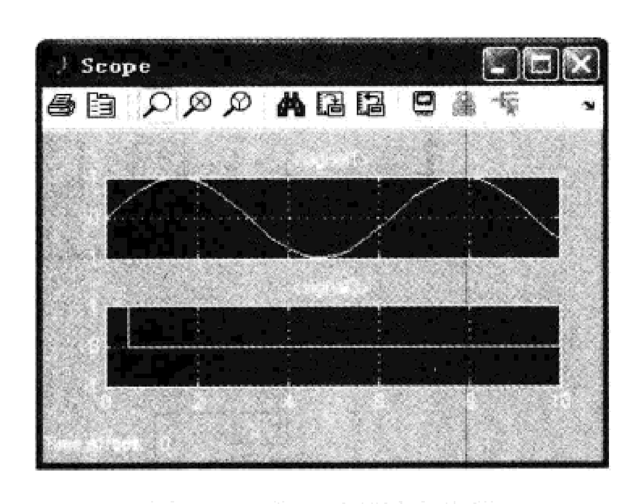


图 3-17　信号选择输出结果

2. 信号合成与信号分离模块

Mux（信号合成器）模块和 Demux（信号分离器）模块的功能看似与总线信号生成模块和总线信号选择模块的功能相似，但是信号合成器与信号分离模块是对所有信号进行合成与分离的，而总线信号选择模块可任意选择总线上的信号进行输出。如图 3-18 所示，双击 Mux 模块，在弹出的参数设置对话框中，将“Number of inputs”参数改为 3，双击 Demux 模块，在弹出的参数设置对话框中，将“Number of inputs”参数改为 3。运行仿真，可得仿真结果如图 3-19 和图 3-20 所示。

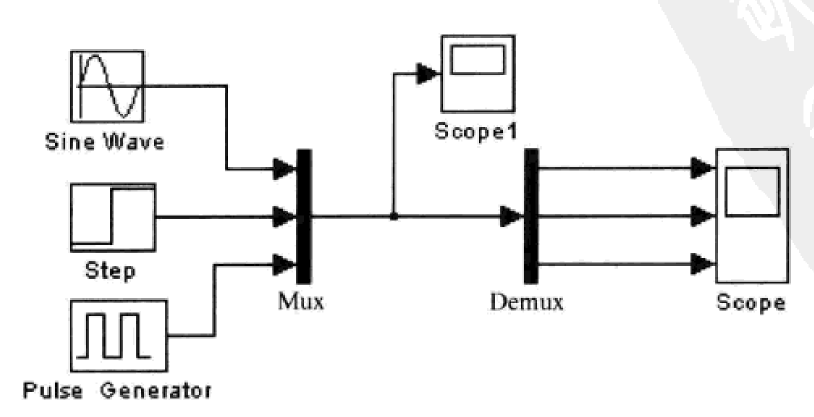


图 3-18　信号合成与分离

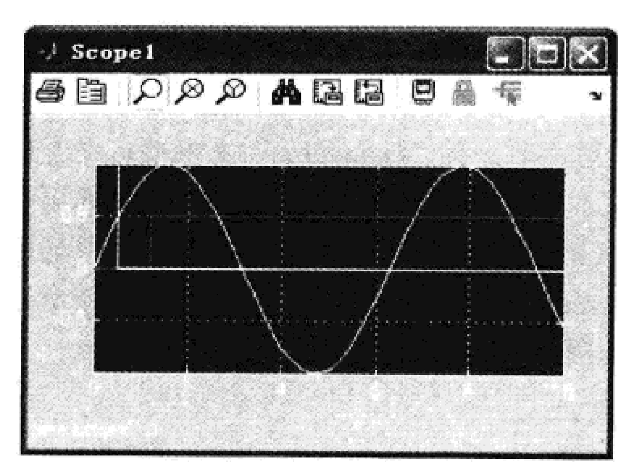


图 3-19　信号合成结果

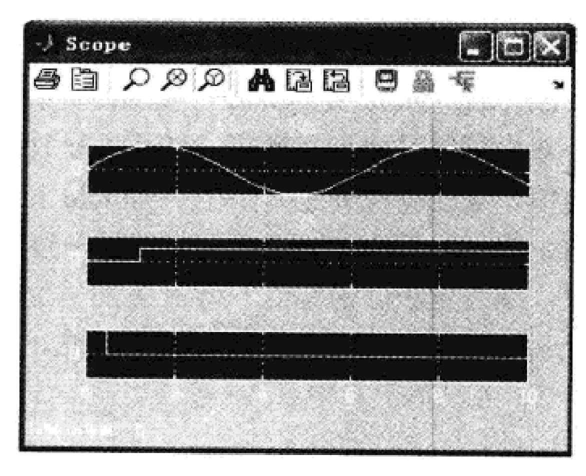


图 3-20　信号分离结果



3．数据类型转换模块

Data Type Conversion（数据类型转换）可将输入数据转换为指定输出类型，具体选择有Inherit（与输入数据保持一致）、Double（双精度类型）、Single（单精度）、int8（8 位整数数据）、uint8（无符号 8 位整型数据）等。在输入/输出数据上可选择 Real World Value（实数值相等）或者 Stored Integer（存储整数相等）。同时可以选择取整的方向，如选择“Round Integer Calculations toward”为“Zero”时，表示向零取整；选择“Nearest”时，表示向量最接近的整数取整；选择“Floor”时，表示向负无穷取整；选择“Ceiling”时，表示向正无穷取整。

4．积分模块

Integrator（积分）模块为连续时间积分单元。双击如图 3-21 所示的 Integrator（积分）模块，打开“参数设置”对话框，在“Initial condition”文本框可设置积分器初始值，在“Limit output”文本框可设置积分器输出最大和最小值。运行仿真后，可得积分器输出结果如图 3-22 所示。

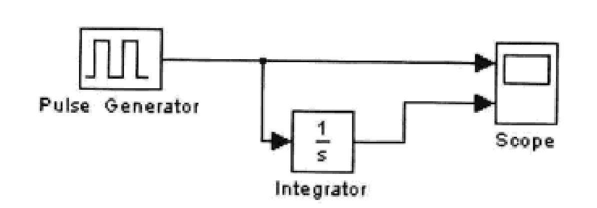


图 3-21　积分器模块示例

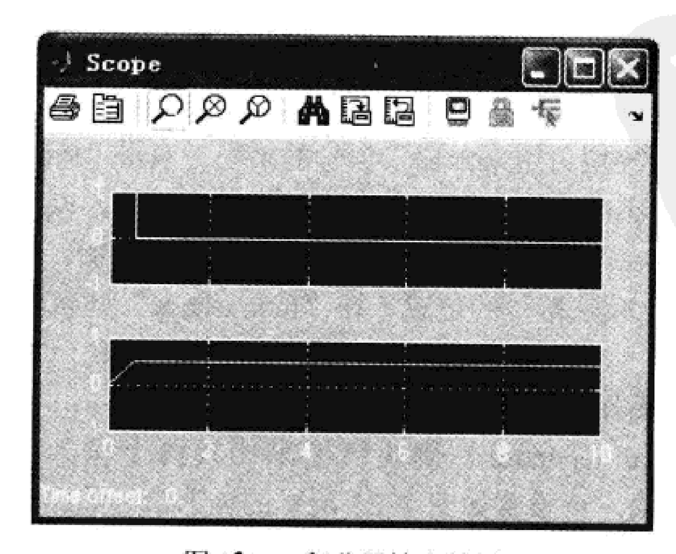


图 3-22　积分器输出结果

5. 离散时间积分模块

Discrete Time Integrator（离散时间积分）模块可完成离散系统积分作用。如图 3-21 所示，双击离散时间积分模块，打开“参数设置”对话框。在“Gainvalue”文本框可设置积分增益值，来改变积分速度。在“Sample time”文本框可设置离散积分的采样时间，如果设置为-1，表示与输入信号采样时间保持一致。在“Limit output”文本框可设置离散积分输出上下限，即设置积分饱和值。将图 3-23 中的“Discrete-Time Integrator1”模块积分增益值设置为 2，将采样时间设置为 0.5，运行仿真，可得如图 3-24 所示的结果。

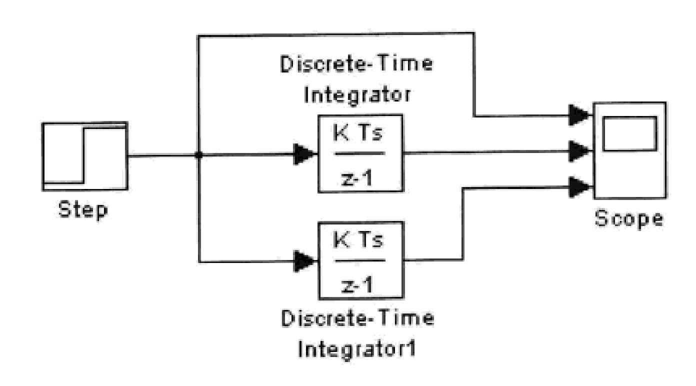


图 3-23 离散时间积分器示例

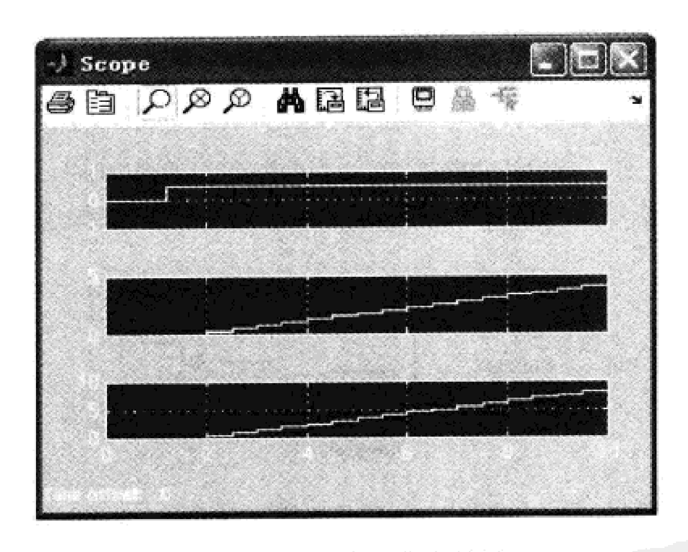


图 3-24 积分器输出结果

6. 乘法与加法模块

“Product”（乘法器）模块可用来求输入信号的乘积，双击“Product”模块可修改输入信号的端口数。“Sum”（加法器）模块可用来求输入信号的加法、减法操作。双击“Sum”模块打开参数设置对话框，在“List of signs”符号列表框可修改加、减法符号，改变符号列表栏“|”位置，可改变加、减法符号的位置，在“Icon shape”列表框中可选择加法器模块的外部形状为圆形或方形。

7. 关系操作及逻辑操作模块

关系操作（Relational Operator）模块可用来比较两个输入信号的大小关系，双击“Relational Operator”模块，可设置如图 3-25 所示的输入信号的比较关系、关系操作可选择大于等于（>=）、小于等于（<=）、不等于（~=）、等于（==）、大于（>）、小于（<）。

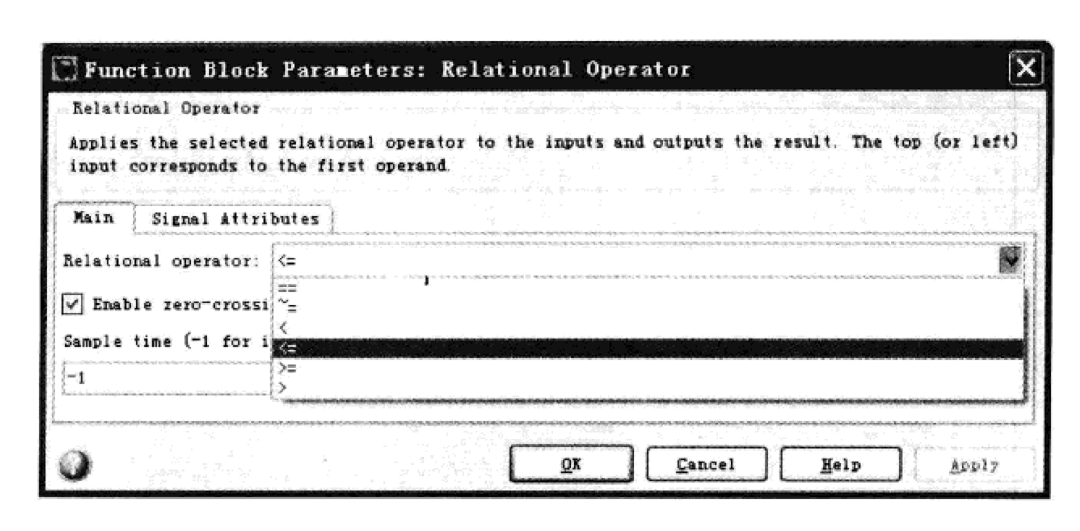




图 3-25 关系操作模块参数设置对话框

逻辑操作（Logic Operator）模块可用来求取两输入变量的逻辑操作关系，双击“Logic Operator”模块，打开如图 3-26 所示的“Logic Operator“模块参数设置对话框，逻辑操作具体有与（AND）操作、或（OR）操作、非（NOT）操作、异或（XOR）操作、与非（NAND）操作、或非（NOR）操作。

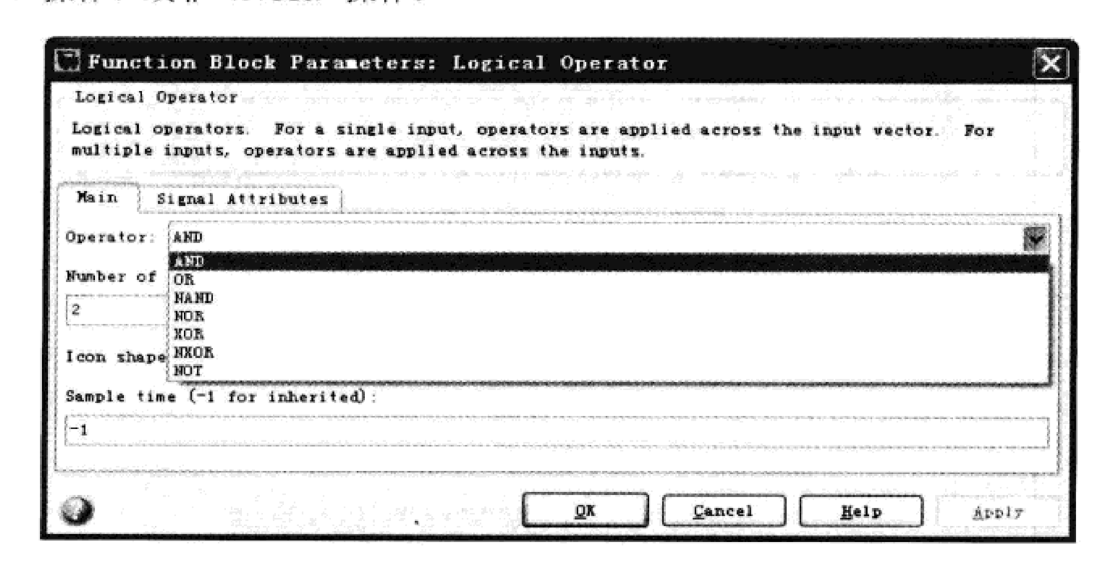


图 3-26 逻辑模块参数设置对话框

8. 增益与饱和模块

Gain（增益）模块可用来设置信号放大倍数，在 Simulink 动态仿真中使用频繁。双击“Gain”模块打开参数设置对话框，在“Gain”（增益）文本框中可设置具体信号放大倍数。

9. 输入/输出接口及子系统模块

In1（输入）模块在建立子系统时作为输入信号的接口。Out1（输出）模块在建立子系统时作为输出信号的接口。Subsystem（子系统）模块可用来将复杂系统的全部或局部生成为一个子系统，这样便于简化 Simulink 模型结构。

10. 终端模块

终端（Terminator）模块可用来连接没有与其他模块相连的输出端口，在 Simulink 模型中，如果有输出端口没有连接，运行仿真时在 MATLAB 窗口将显示警告信息，使用终端模块可以避免这类警告信息的出现。

3.4.2 连续系统模块库

Continuous（连续系统）模块库提供了连续系统 Simulink 建模与仿真的基本模块，有 Derivative（微分环节）模块、Integrator（积分环节）模块、状态空间（State-Space）模块、Transfer Fun（传递函数功能）模块、Transport Delay（传输延迟）模块、Veriable Transport Delay（可变时间延迟）模块、Veriable Transport Delay（可变传输延迟）模块和 Zero-Pole（零极点增益）模块。打开独立连续系统模块库，如图 3-27 所示。

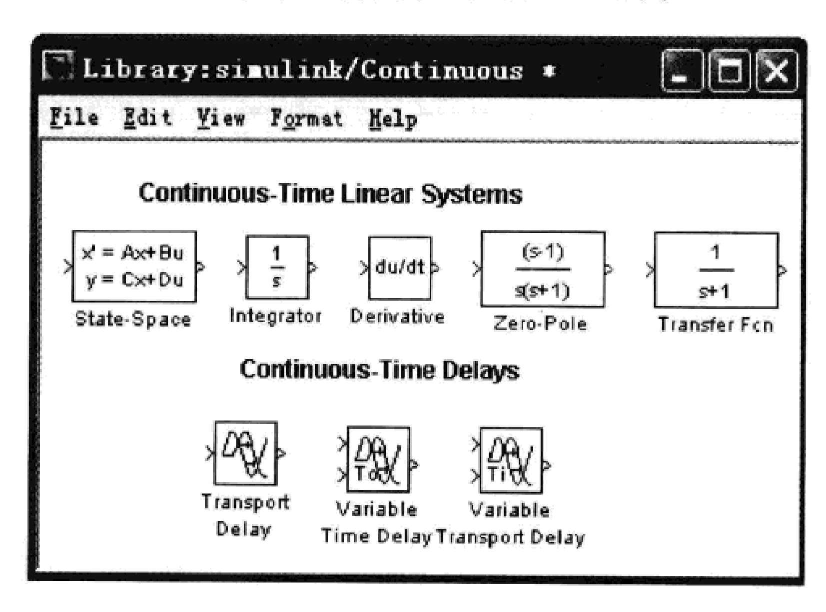


图 3-27　连续系统模块库

1．状态空间模型

使用 Simulink 动态仿真模型，可使用状态空间 State-Space 模块，双击“State-Space”模块，系统弹出如图 3-28 所示的“参数设置”对话框，在这里可设定状态空间模型的系数矩阵及初始状态。

图 3-28　状态空间模型“参数设置”对话框



2. 微积分模块

使用微分（Derivative）模块、积分（Integrator）模块可建立各种各样的控制器及控制系统动态仿真模型。双击“Derivative”模块，打开如图 3-30 所示的微分环节线性化设置对话框。系统默认的微分环节线性化时间常数为 inf（无穷大）。如果微分单元数学模型为 $T_d s$，则当 $N \geqslant 20$ 时，$\dfrac{T_d s}{\dfrac{T_d s}{N}+1}$ 可近似代替 $T_d s$。所以在图 3-29 中，用户可根据需要自行设定 N 的数值。



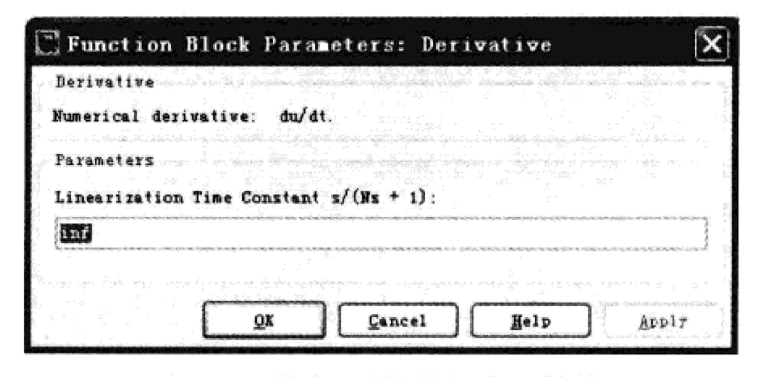


图 3-29 微分环节线性化设置对话框

3. 传输延迟及可变传输延迟模块

如果被控系统模型里含有纯延迟环节，用户可以使用 Simulink 工具箱连续系统模块库提供的“Transport Delay”模块来建立仿真模型。双击“Transport Delay”模块，系统弹出“参数设置”对话框，在“Time Delay”文本框中输入需要延迟的时间数值。同时，在“Pade Order”文本框中用户可以输入纯延迟环节线性化处理的近似多项式阶数。

Veriable Time Delay（可变时间延迟）模块与 Variable Transport Delay（可变传输延迟）模块在 Simulink 连续系统模块库里以两个模块的形式出现，但是它们可以通过选择模型属性的“Select Delay Type”属性值来相互变换。

传输延迟模块应用在传输的延迟时间与被控对象传输速度有关的系统建模上。例如，液体在管道中传输，液体的流速随时间变化，则液体在管道中的传输时间（即延迟时间）就是一个与液体传输速度有关的变量。假定管道长度 L 一定，若对某部分液体添加标记，则该部分液体通过管道的时间（即管道的传输延迟）与液体在管道中的流速之间的关系为：

$$\tau_i(t)=\frac{L}{v_i(t)}$$

4. 传递函数模型与零极点增益模型

传递函数（Transfer Fcn）模块可以用来建立连续系统传递函数的 Simulink 仿真模型，双击“Transfer Fcn”模块，打开如图 3-30 所示的“参数设置”对话框。在图 3-30 中，在“Numerator coefficient”文本框中定义分子多项式系数向量，在“Denominator coefficient”文本框中定义分母多项式系数向量，从而建立传递函数的 Simulink 仿真模型。

如果用户已知系统的零点、极点和增益，也可以使用连续系统的零极点增益模型来建立 Simulink 仿真模型。需要修改传递函数模型时，只需打开零极点增益模型的参数设置对话框，在指定的零点、极点、增益文本框中输入相应的数据，然后单击“OK”按钮即可完成零极点增益模型的 Simulink 建模。

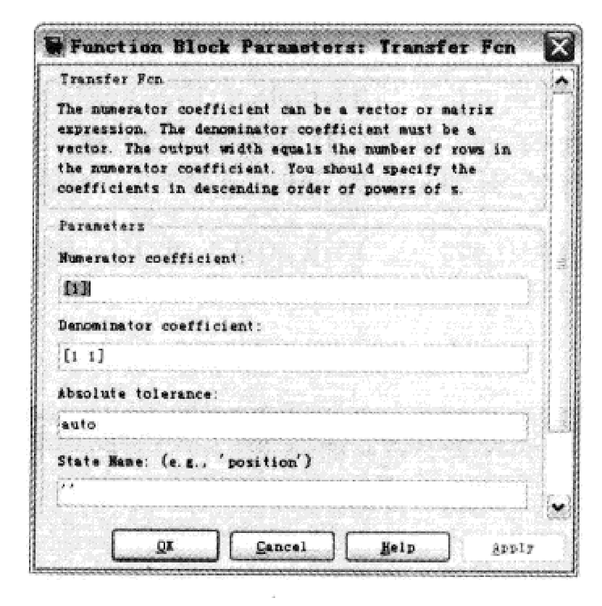


图 3-30　增益模块参数设置对话框

【例 3-1】 建立一个连续系统仿真模型，包括延迟及可变延迟单元。

根据上述连续系统模块库的讲解，建立仿真模型，如图 3-31 所示。输入为正弦波信号，固定延迟时间设为 2s，可变延迟模块延迟时间最大值为 10，延迟时间由阶跃信号确定，初始值设为 3，终止值设为 5，阶跃时间为 10s。可变传输延迟模块延迟时间定义为 2 除以当前输入信号值。运行仿真后，输出效果如图 3-32 所示。

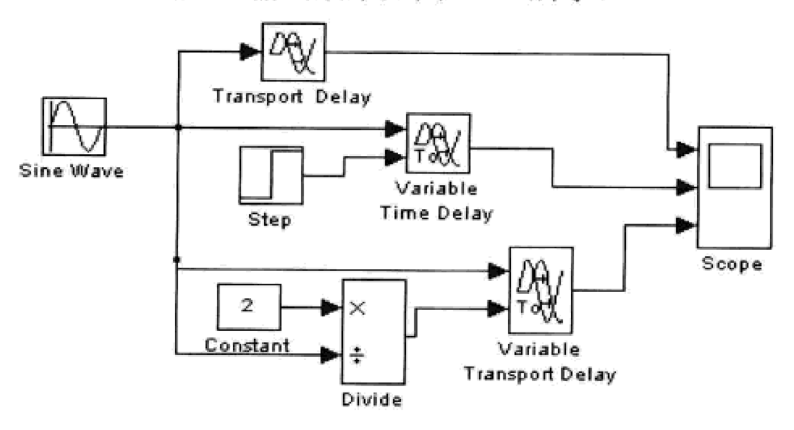


图 3-31　传输延迟模块仿真示例

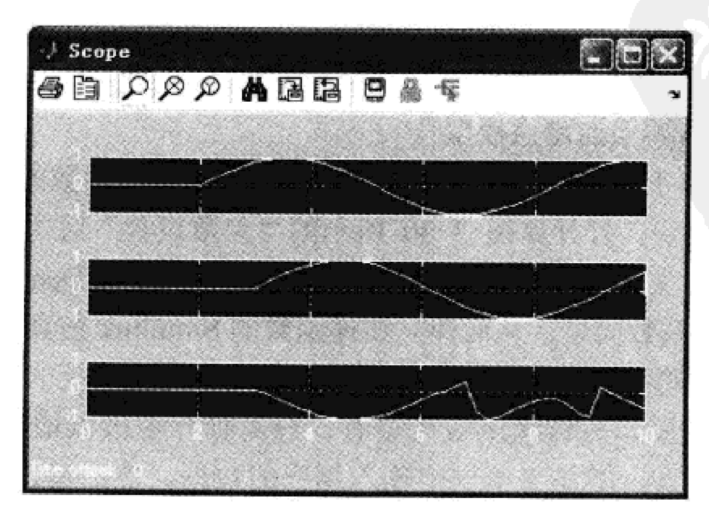


图 3-32　仿真结果

3.4.3 非连续系统模块库

Discontinites（非连续系统）模块库在以前版本中也称为非线性模块库，包含一些常用的非线性运算模块，单独的非连续系统模块库如图 3-33 所示。该模块主要包括以下内容。



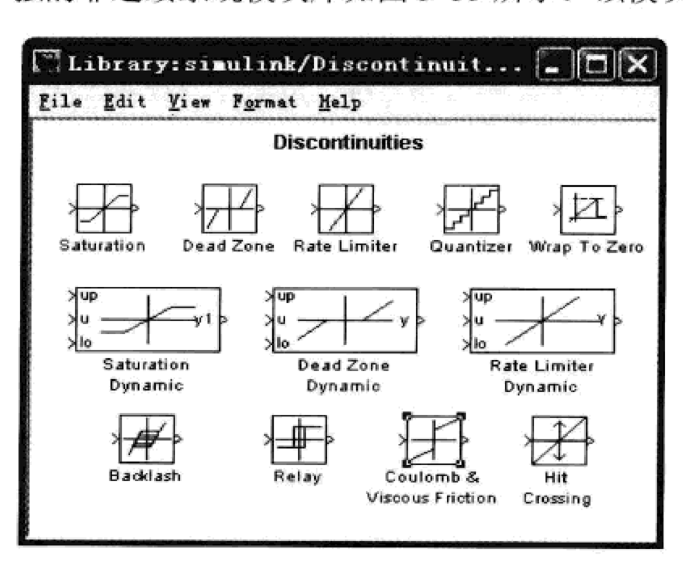


图 3-33　非连续系统模块库

（1）Saturation（饱和度）模块

该模块对一个信号限定上下限。当输入在由 Lower limit 和 Upper limit 参数指定的范围内时，输入信号无变化输出。若输入信号超出范围，则信号就会被限幅（值为上限或下限）。若这两个参数的设置值相等时，模块就输出该值。模块只接受和输出双精度实型信号。Saturation Dynamic（动态饱和非线性）模块可以根据输入端口 Up 和 Lo 的设定值动态设置输出的上限和下限。

（2）Dead Zone（死区模块）模块

该模块提供了一个死区特性，即产生在指定范围内的零输出。模块用 Start of dead zone 和 End of dead zone 参数指定截止区的下限值和上限值。模块的输出取决于输入和截止区的大小：

- 若输入落在截止区内（大于下限值且小于上限值），则输出为 0。
- 若输入大于或等于上限值，则输出等于输入减去上限值。
- 若输入小于或等于下限值，则输出等于输入减去下限值。

（3）Rate Limite（速度限制）模块

该模块用于限定通过模块的信号的一阶导数，以使输出的变化不超过指定界限。导数根据方程 $rate=\dfrac{u(i)-y(i-1)}{t(i)-t(i-1)}$ 计算得出。其中，$u(i)$ 和 $t(i)$ 为当前模块的输入和时间，$y(i-1)$ 和 $t(i-1)$ 为前一时间步的输出和时间。

（4）Quantizer（量子点）模块

该模块对输入信号进行量化处理。将平滑的输入信号变为阶梯状输出。输出计算采用四舍五入法，产生与零点对称的输出。

$$y = q * round(u / q)$$

式中，y 为输出；u 为输入；q 为 Quantization interval 参数。

（5）Wrap To Zero（限零）模块

当输入信号超过 Threshold 参数限定值时，模块产生零输出；当输入信号小于或等于限定值时，输入信号无变化输出。

（6）Backlash（磁滞回环）模块

该模块可实现输入和输出变化相同的系统。然而，当输入改变方向时，输入的初始变化对输出没有影响。在系统中，这个四边形的区域称为回差或死区（Deadband）。此死区的中点就是输出信号的原点。

（7）Coulomb & Viscous Friction（库仑和黏性摩擦）模块

建立库仑（静）力和黏滞（动）力模型。该模块建立的是在零点不连续而其余点线性的增益模型。偏置对应库仑力，增益对应黏滞力。该模块由以下的函数式表示：

$$y = \mathrm{sign}(u) * (\mathrm{Gain} * \mathrm{abs}(u) + \mathrm{Offset})$$

其中，y 代表输出；u 代表输入；Gain 和 Offset 为模块参数。

（8）Dead Zone Dynamic（动态死区）模块

该模块动态限制输入信号的范围，产生在指定范围内的输出死区。模块的限制范围取决于输入信号的上限值和下限值：

- 若输入落在限制范围内，则输出为 0。
- 若输入大于上限值，则输出减小到上限值。
- 若输入小于下限值，则输出增大到下限值。

（9）Hit Crossing（捕获穿越点）模块

该模块用于将输入信号与 Hit Crossing Offset 参数值进行比较。当输入信号超过参数值时，若输入等于或低于参数值，模块输出为 1，否则输出为 0。

（10）Rate Limiter Dynamic（动态限速）模块

该模块用来限制信号的递增速率和递减速率，使其不超过规定的限制值。

（11）Relay（继电器）模块

在两个值中轮流输出。当模块状态为“on”时，此状态一直保持到输入下降到比 Switch off point 参数值小。若为“off”，此状态一直保持到输入超过 Switch on point 参数值。模块接受一个输入，产生一个输出。

3.4.4 离散系统模块库

离散系统模块库主要包括用于建立离散采样系统的模块。单独的离散系统模块库如图 3-24 所示。离散系统模块库主要包括以下内容。

（1）Unit Delay（单位延迟）模块

该模块延迟一个采样周期。

（2）Integer Delay（整数延迟）模块

该模块延迟输入 N 个采样周期，N 为自然数。

（3）Tapped Delay（触发延迟）模块

该模块延迟 N 个采样周期后输出全部的输入信息。

（4）Discrete Transfer Fcn（离散传递函数功能）模块

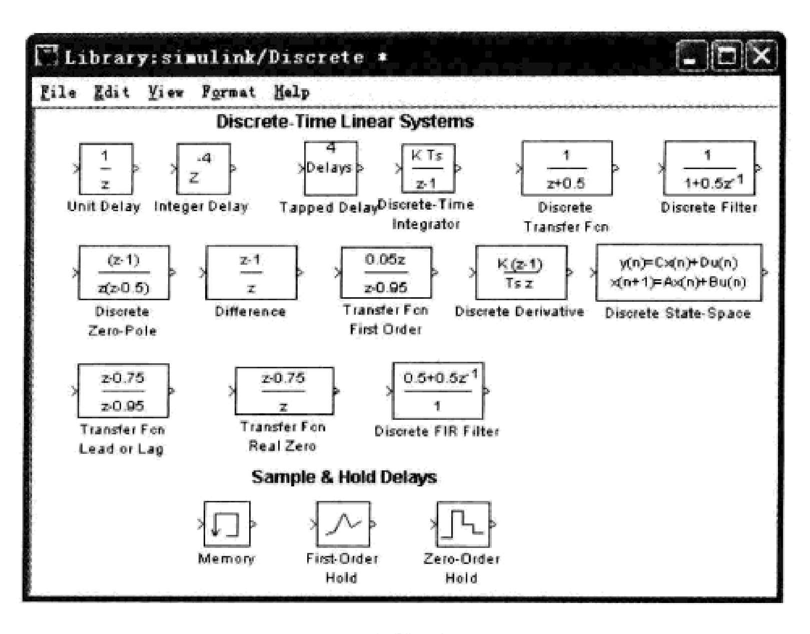




图 3-34 离散系统模块库

该模块可以建立离散传递函数模型。

（5）Discrete Filter（离散过滤分析）模块

该模块用于实现无限冲激响应（IIR）和有限冲激响应（FIR）滤波器。用户可以用 Numerator 和 Denominator 参数指定以 z^{-1} 的升幂为矢量的分子和分母多项式的系数。分母的阶数必须大于或等于分子的阶数。

（6）Discrete State-Space（离散状态空间）模块

离散状态空间模块实现如下一个离散系统：

$$x(n+1) = \boldsymbol{A}x(n) + Bu(n)$$
$$y(n) = \boldsymbol{C}x(n) + \boldsymbol{D}u(n)$$

式中，u 为输入；x 为状态；y 为输出。矩阵系数必须满足以下要求：

- $\boldsymbol{A}$ 必须是 $n \times n$ 矩阵，其中 n 为状态数量。
- $\boldsymbol{B}$ 必须是 $n \times m$ 矩阵，其中 m 为输入数量。
- $\boldsymbol{C}$ 必须是 $r \times n$ 矩阵，其中 r 为输出数量。
- $\boldsymbol{D}$ 必须是 $r \times m$ 矩阵。

模块接受一个输入，并产生一个输出。输入矢量宽度由矩阵 $\boldsymbol{B}$ 和 $\boldsymbol{D}$ 的列数决定。输出矢量宽度由矩阵 $\boldsymbol{C}$ 和 $\boldsymbol{D}$ 的行数决定。Simulink 将一个包含 0 的矩阵转换成一个利于相乘的稀疏矩阵。

（7）Discrete-Time Integrator（离散时间变量积分）模块

当用于构建纯离散系统时，该模块可替代 Integrator 模块，允许用户完成下列任务：

- 在模块的对话框中定义初始条件或模块的输入。
- 输出模块状态。
- 定义积分上下限。
- 根据附加位置输入重新设置状态。

（8）First-Order Hold（首要控制）模块

该模块在指定的时间间隔实现一阶采样保持，主要用于理论研究。模块输入和输出双精度信号。

（9）Zero-Order Hold（零点控制）模块

该模块用于实现一个以指定采样率的采样与保持操作。模块接受一个输入，并产生一个输出，两者可以是标量或矢量。

（10）Fransfer Fcn First Order（一阶离散传递函数功能）模块

该模块用于建立一阶的离散传递函数模型。

（11）Fransfer Fcn Lead or Lag（传递函数引导或终止）模块

该模块用于实现输入信号的离散时间引导或终止补偿。

（12）Fransfer Fcn Real Zero（实数零点传递函数功能）模块

该模块用于实现具有一个实数零点而无极点的传递函数模型。

（13）Memory（记忆）模块

该模块输出前一时刻的输入信号值。

（14）Discrete Derivative（派生离散微分）模块

模块的输出值按下式计算：

$$y(t_n) = K_n \frac{u(t_n) - u(t_{n-1})}{T_n}$$

式中，$y(t_n)$、$u(t_n)$为当前时刻的输入、输出值；$u(t_{n-1})$为前一采样时间的输入值；K_n为引入的比例系数；T_n为仿真的离散步长。

（15）Discrete Filter（离散滤波器）模块

该模块用于建立离散系统滤波器仿真模型。

（16）Difference（离散微分）模块

该模块输出当前输入信号值与前一个采样值之差。

（17）Weighted Moving Averge（平均加权滑动）模块

该模块模拟采样并保持最近 N 个输入信号，根据设定的权值参数值（Weights）计算它们的平均值。该模块适用于 SISO 系统或 SIMO 系统模型。

本模块的工作原理是：离散化一个或多个信号，或以不同的采样率对信号进行重新采样。用户可以将模块用于需要构建采样但不要求其他更复杂的离散函数模块的场合。例如，可将本模块与 Quatizer 模块联合使用，以构建一个单输入 A/D 变换器。

3.4.5 逻辑与位操作模块库

Logic and Bit Operations（逻辑与位操作）模块库提供了建立逻辑系统及数字系统 Simulink 仿真建模的基本模块。打开独立逻辑与位操作模块库如图 3-35 所示。因控制系统动态仿真用到该模块内容较少，在这里就不作介绍。

3.4.6 数学操作模块库

Math Operations（数学操作）模块库提供了与数学运算相关的 Simulink 仿真模块，打开独立的数学模型库，如图 3-36 所示。数学模块库主要包括以下内容。

（1）Sum（求和）模块、Add（加法）模块、Subtract（减法）、Sum of Elements（元素求和）模块

这几个模块通过参数设置，都可以实现加、减运算，而且能够对标量、向量、矩阵输入

数据进行加、减操作。

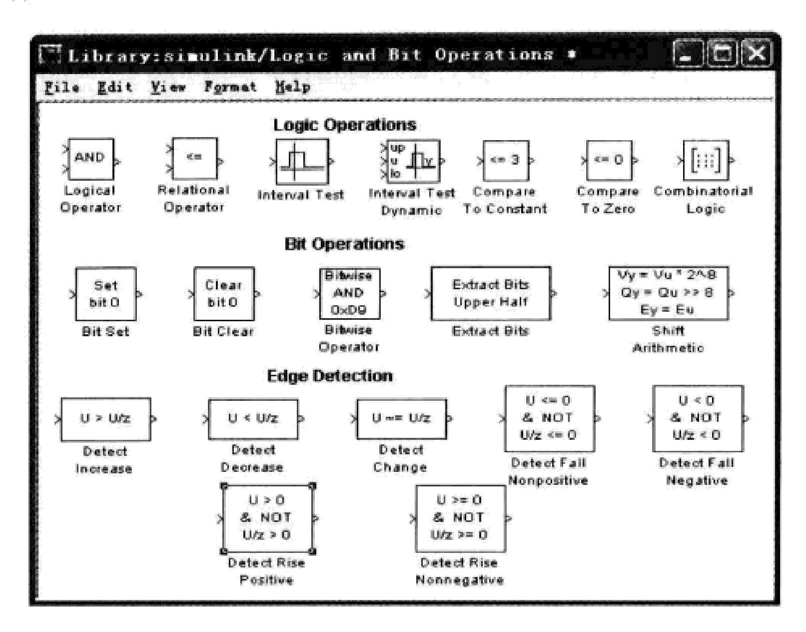


图 3-35 逻辑与位操作模块库

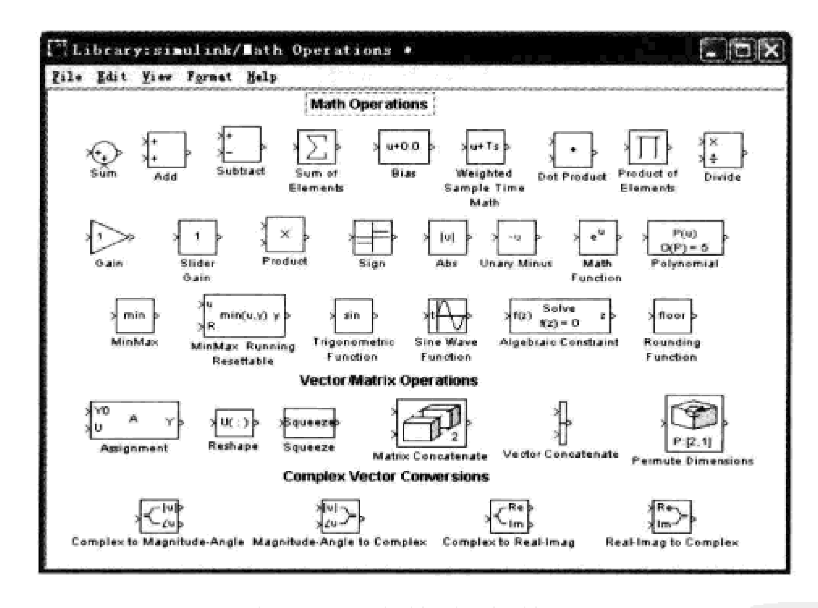


图 3-36 数学操作模块库



（2）Bias（偏差）模块

该模块用于将输入量加上偏差。所依据的公式为：

$$Y = U + Bias$$

式中，U 为模块输入；Y 为输出；$Bias$ 为偏差。

（3）Weighted Sample Time Math（加权采样时间数学操作）模块

该模块可用输入信号加、减、乘或除以 $T_s \times w$，$T_s \times w$ 是 T_s（采样时间）乘以 Weight（加权系数）。

（4）Gain（增益）模块

增益模块将模块的输入乘上一个指定的常数、变量或表达式后输出。用户可以输入数值或变量或表达式。如果模块足够大，Gain 的图标显示 Gain 参数域中输入的值。如果是变

量，则显示变量名。如果 Gain 参数过长，则显示-K-。

（5）Dot Product（点积）模块

该模块对两个输入矢量进行点积运算。标量输出 $y = \boldsymbol{u}1 \cdot \boldsymbol{u}2$ 。其中， $\boldsymbol{u}1$ 和 $\boldsymbol{u}2$ 表示矢量输入。若两个输入均为矢量，长度必须相等。输入矢量的元素可以是双精度实数或复数信号。输出信号的数据类型取决于输入。

（6）Product（乘积）模块

该模块对输入进行乘法或除法运算。模块接受任意数据类型的实/复信号。所有输入信号的数据类型必须相同，所以输出信号的数据类型也与输入相同。

（7）Magnitude-Angle to Complex（幅值和幅角转换为复数）模块

该模块能将一个幅度和（或）一个相角信号变换为复信号输出。输入必须是双精度型实信号。相角单位假设为弧度。复信号输出是双精度型。输入可以是两个相同大小的矢量，或者一个是矢量，另一个是标量。如果模块的输入是矢量，则输出是复信号矢量。一个幅度输入矢量的元素映射到对应复输出信号的幅度。类似地，一个相角输入矢量映射到复输出信号的相角。若一个输入是标量，则它映射到所有复输出信号的对应成分（幅角或相角）上。

（8）Math Function（数学函数）模块

该模块可以进行多种常用数学函数运算。用户可以从 Function 表中选择如下函数之一：exp、log、10u、log10、square、sqrt、power 等。模块输出用这些函数对输入进行计算以后的结果。函数名在模块上显示。

（9）Rounding Function（环绕取舍函数）模块

该模块用于实现常用的数学取整函数。模块接受和输出双精度型实或复信号。

（10）Polynomial（多项式）模块

该模块用于显示输入量的多项式系数。

（11）MinMax（最大与最小）模块

该模块将输入的最小或最大值的元素或所有元素作为输出，用户可以通过选择 Function 参数表中的函数来确定欲使用的函数。若模块有一个输入端口，模块将输入矢量的最小值元素或最大值元素用一个标量输出。若模块的输入端口多于一个，模块将输入矢量的各元素逐一进行比较，模块输出矢量的各元素即为输入矢量各元素的比较结果。

3.4.7 表格查询模块库

Lookup Tables（表格查询）模块库可以用来建立一维、二维或多维表格查询的 Simulink 仿真模型。打开独立的表格查询模块库，如图 3-37 所示。其中主要包括 Lookup Table（基本一维表格查询）模块、Lookup Table（2-D）（二维表格查询）模块、Lookup Table（n-D）（多维表格查询）模块等。

【例 3-2】 试建立一个二维表格查寻的 Simulink 仿真模型。

建立二维表格查询的 Simulink 仿真模型如图 3-38 所示。双击“Sine Wave”模块，系统弹出参数设置对话框，其设置如图 3-39 所示。双击“Rounding Function”模块，系统弹出参数设置对话框，其参数设置如图 3-40 所示。双击“Lookup Table(2-D)”模块，系统弹出参数设置对话框，如图 3-41 所示。在“Row index input value”文本框中输入行索引值，在“Column index input values”文本框输入列索引值。在“Table data”文本框

输入表格元素值。单击“Edit”按钮，系统弹出表格编辑窗口，如图 3-42 所示。在图 3-42 所示的表格编辑窗口中，单击“Edit”菜单下的“Add Row”、“Add Coumn”、“Remove Row(s)”、“Remove Column(s)”命令，可以添加或删除行或列。在图 3-42 的右侧窗格中可以编辑表格内容或数据类型。运行仿真可得输入的行、列下标值及查询结果，如图 3-43 所示。



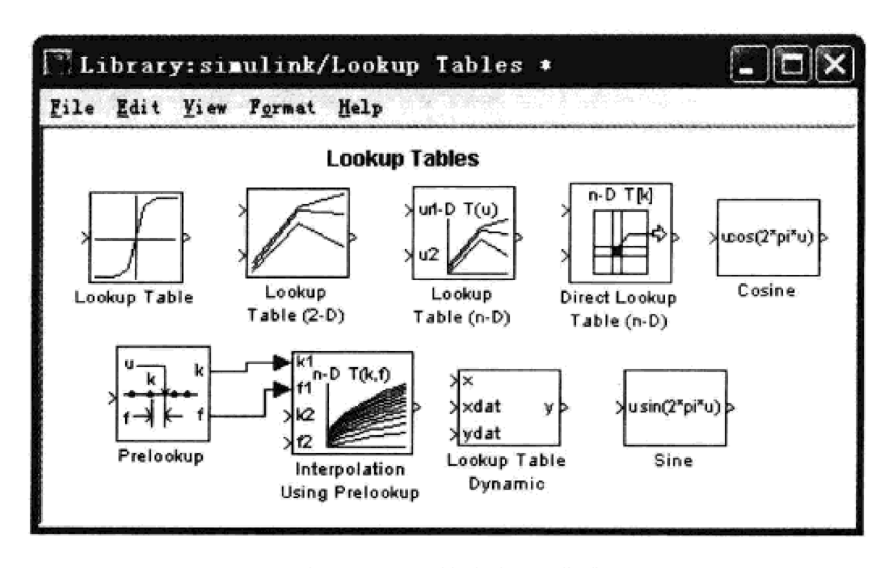


图 3-37 表格查询模块库

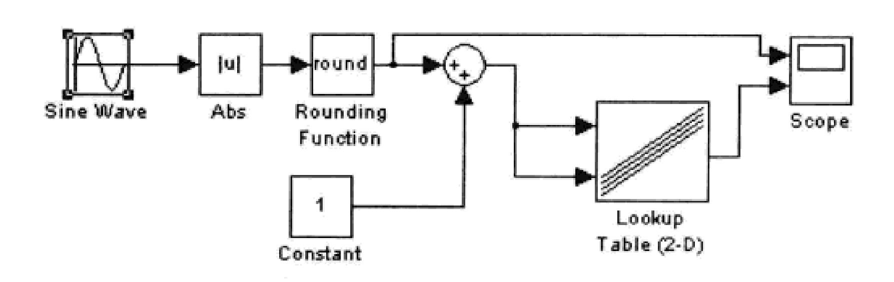


图 3-38 二维表格查询 Simulink 仿真模型

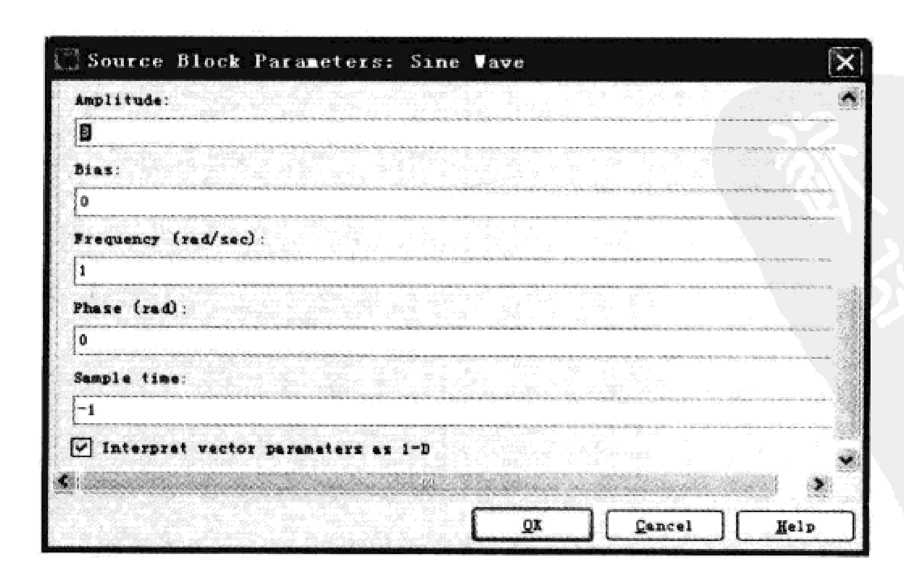


图 3-39 Sine Wave 模块参数设置对话框

图 3-40　Rounding Function 模块参数设置对话框

图 3-41　Lookup Table(2-D)模块参数设置对话框

图 3-42　表格编辑窗口

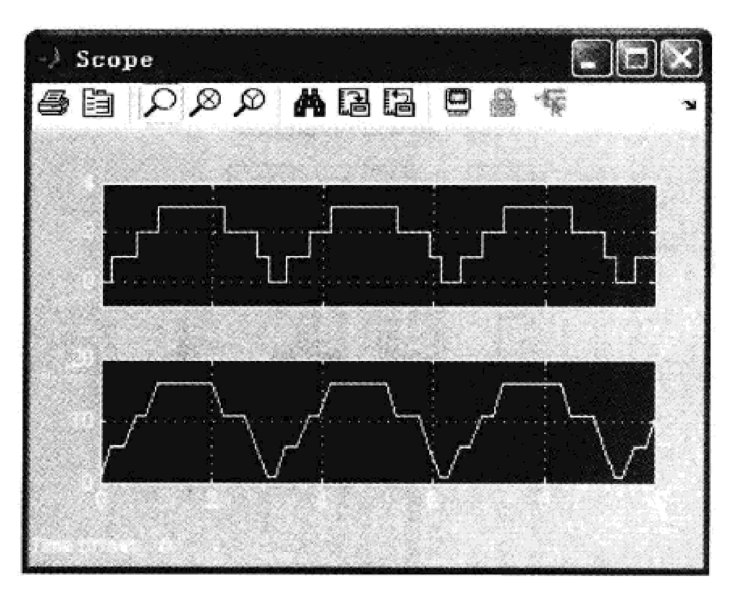


图 3-43 表格查询 Simulink 仿真结果



3.4.8 端口与子系统模块库

Ports & Subsystems（端口与子系统）模块库用于创建各类子系统模型，打开独立端口与子系统模块库如图 3-44 所示。子系创建基本方法将在第 4 章展开介绍，这里先不作介绍。

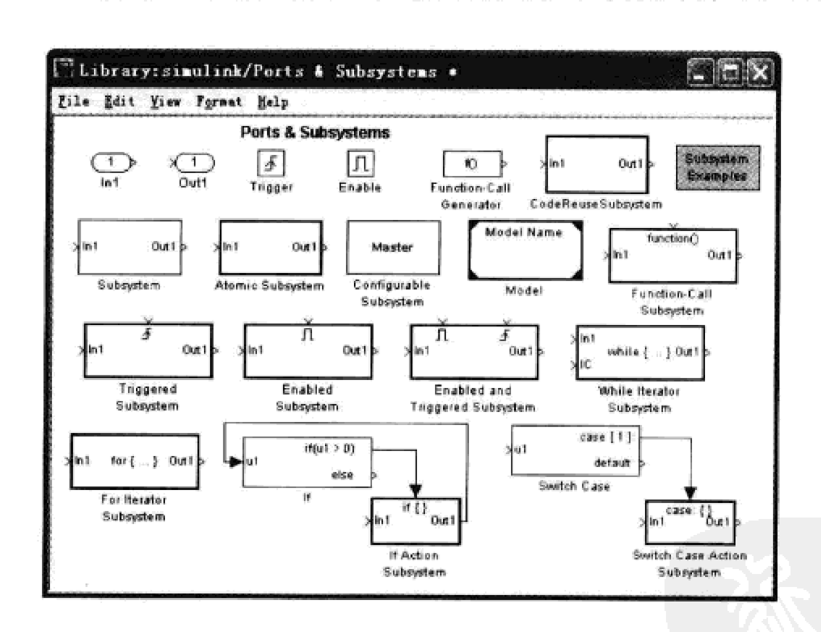


图 3-44 端口与子系统模块库

3.4.9 信号属性操作模块库

Signal Attributes（信号属性）操作模块库可用于信号系统的 Simulink 建模与仿真。这里不多作介绍，其独立的显示窗口如图 3-45 所示。

3.4.10 信号路由模块库

Signal Routing（信号路由）模块库用于控制信号的传递路径。打开单独信号路由模块库，如图 3-46 所示。其主要应用模块如下：

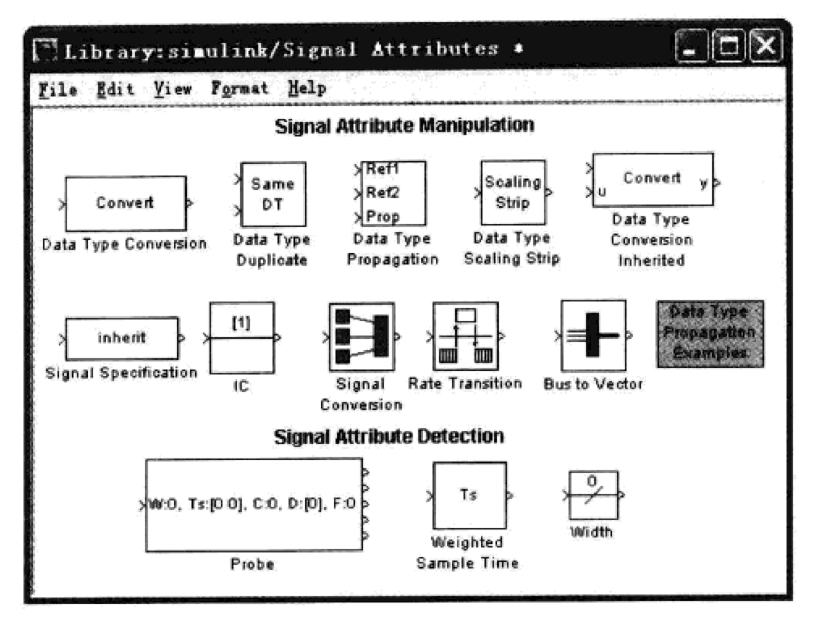


图 3-45　信号属性操作模块库

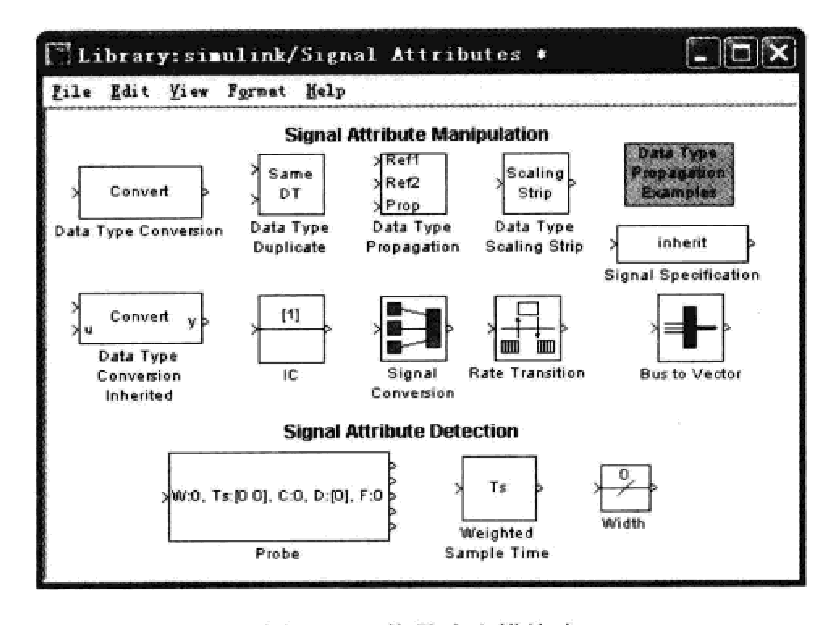


图 3-46　信号路由模块库

（1）Bus Creater（总线生成器）、Bus Selector（总线选择器）、Bus Assignment（总线分配器）

总线生成器用于生成信号总线，总线选择器用于从输入的总线上选择信号，总线分配器分配用于替代指定的信号单元。

（2）Data Store Read 模块、Data Store Memory 模块、Data Store Write 模块

这几组模块分别用于将存储的数据读入到内存空间、将数据存储到内存空间、将数据写入到存储的数据文件中。

（3）From 模块、Goto 模块、Goto Tag Visibility 模块

这几个模块分别用于从工作空间变量输入信号、将信号输出到变量、将信号输出到可见的变量。

（4）Manual Switch（手动开关）、Multiport Switch（多路开关）模块

这两个模块可在模块输入信号之间进行选择，第一个输入端口为控制端口。Switch（开关）

模块根据第二个输入信号与 Threshold（模块设定极限值）的比较结果决定选择哪路输入信号通过。当比较结果大于零时，第一个输入端口的信号通过模块输出；当比较结果小于零时，第三个输入端口的信号通过模块输出。



（5）Mux（信号混合器）、Demux（信号分离器）

信号混合器用于合并几个输入信号为一个向量信号。信号分离用于释放并输出向量或总线信号的元素。

（6）Selector（选择器）模块

该模块用于从向量、矩阵或多维信号中选择输入元素，Index Vector 模块根据第一个输入信号值在不同的输入值之间切换输出。

（7）Merge（合并）模块

该模块用于合并多重信号到一个信号。

【例 3-3】 用信号路由模块库及其他模块建立 Simulink 仿真模型。

利用信号路由模块库建立的 Simulink 仿真模型如图 3-47 所示。Pulse Generator 模块的参数设置如图 3-48 所示。Sine Wave 的参数设置如图 3-49 所示。其他参数采用默认设置。

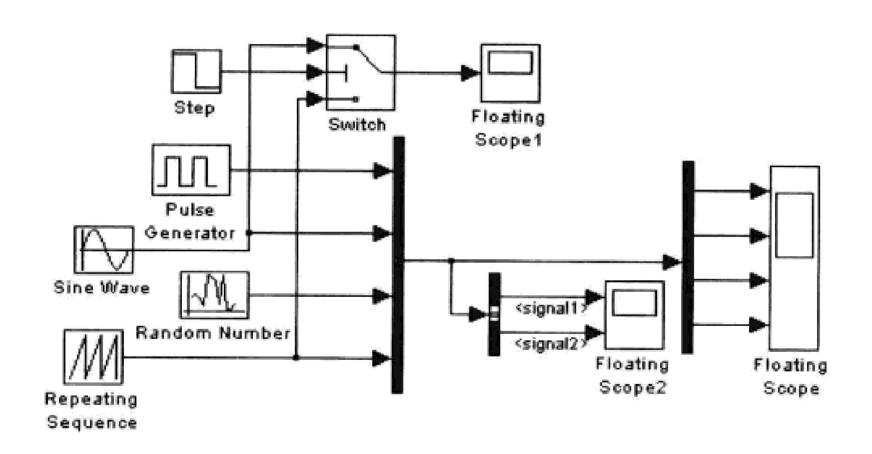


图 3-47　仿真模型

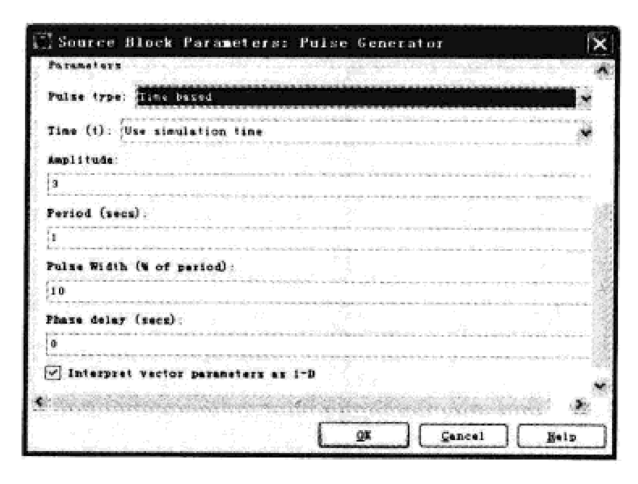


图 3-48　Pulse Generator 参数设置对话框

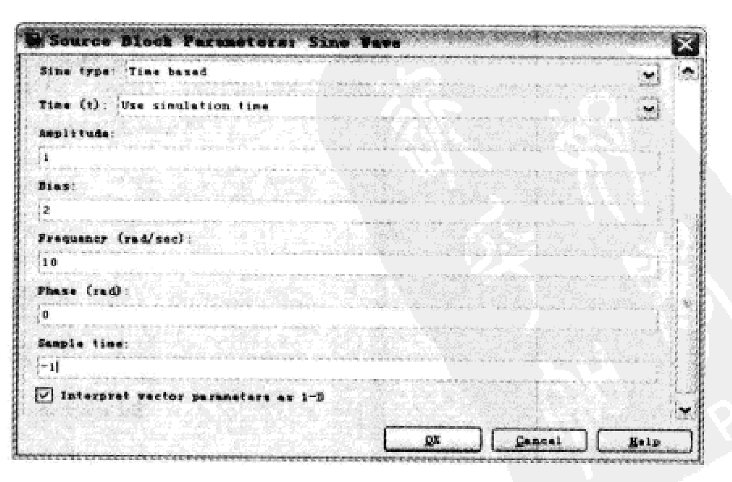


图 3-49　Sine Wave 参数设置对话框

参数设置完毕后，运行仿真模型，可得如图 3-50～图 3-52 所示的结果。

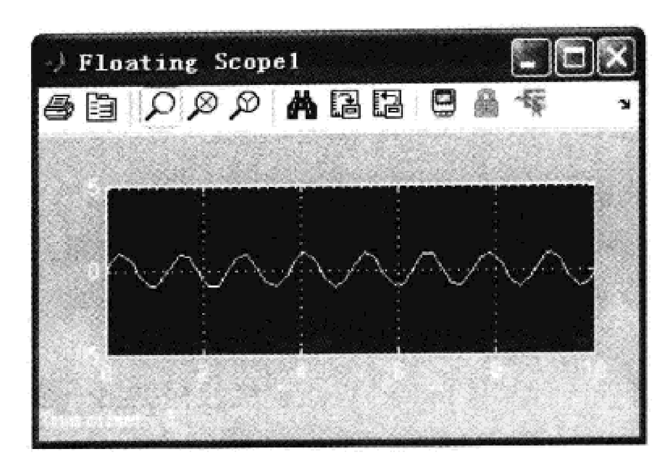


图 3-50　开关模型输出结果

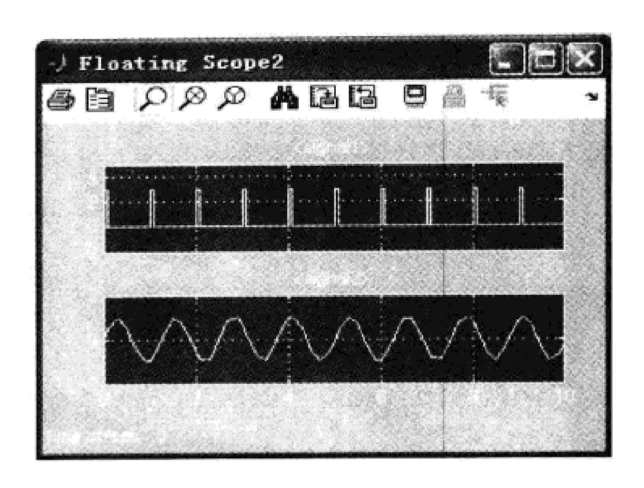


图 3-51　总线选择器输出信号结果

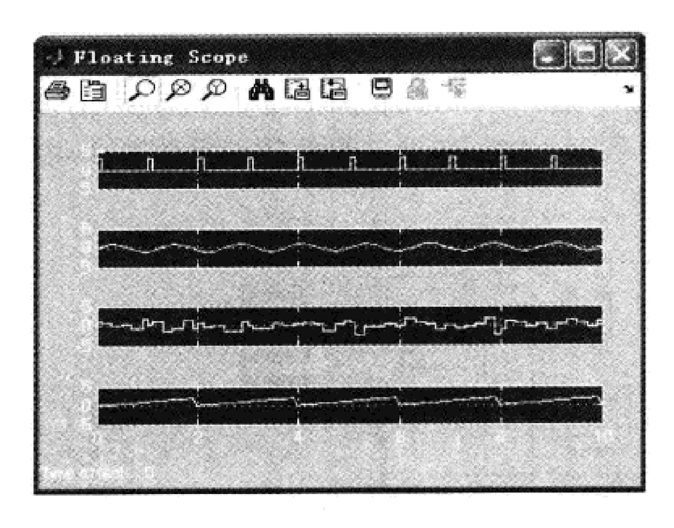


图 3-52　信号分离器输出结果

3.4.11 接收模块库

Sinks（接收）模块库中主要包括 Model & Subsystem Outputs（模块与子系统输出）、Data Viewers（数据观测器）和 Simulation Control（仿真终止模块）3 部分，如图 3-53 所示。

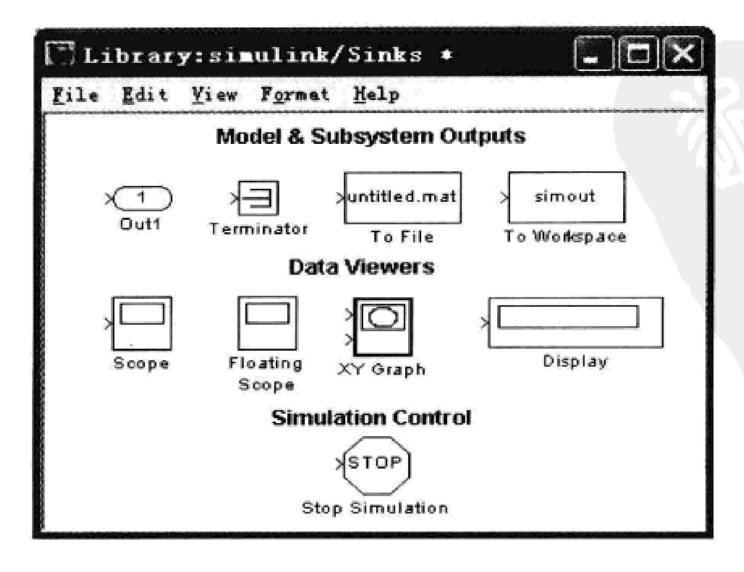


图 3-53　接收模块库

其模块与子系统输出模块主要包括以下内容。

（1）Out1（子系统输出）模块

该模块用于子系统输出端口。

（2）Terminator（信号终端）模块



当某输出信号没有其他模块与其相连接时，可使用 Terminator 模块；否则将在 MATLAB 命令窗口提示警告信息。

（3）To File 模块

该模块将数据输出到文件。

（4）To Workspace 模块

该模块该模块将与其相连接的数据输出到工作空间，双击该模块弹出参数设置对话框，如图 3-54 所示。在这里可以修改将数据输出到工作空间的变量名称“Variable name”，同时还可以在“Save format”下拉列表框中设置修改数据存储格式。通常可选择 MATLAB 常用的数据存储格式“Array”。

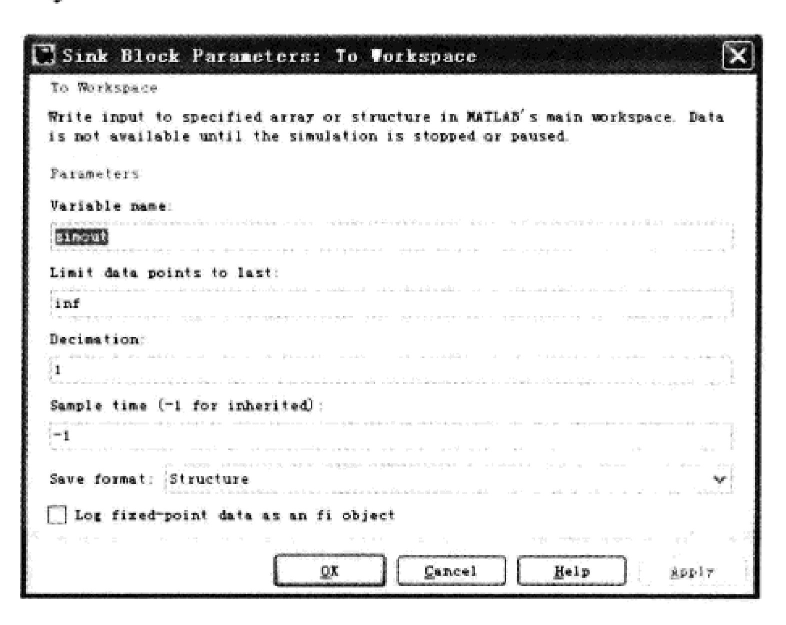


图 3-54　To Workspace 模块参数设置对话框

数据观测模块库和仿真终止模块主要包括以下内容。

（1）Scope（示波器）模块

以 Simulink 仿真时间为横坐标，以示波器端口输入信息为纵坐标绘制图形。双击示波器可打开图形显示及参数设置对话框。

（2）Floating Scope（浮动示波器）模块

浮动示波器没有固定的输入端口，这样可减少模型窗口的连接线，使窗口简洁。在使用浮动示波器时，需要首先设置信号。其参数设置与 Scope 参数设置一样。

（3）XY Graph（XY 轴双输入示波器）模块

该模块有两个输入端口，以第一个输入端口为 X 轴坐标，以第二个端口输入为 Y 轴坐标绘制图形。双击该模块可打开模块参数设置对话框。

（4）Display（数字显示波器）模块

该模块以数字形式显示当前输入的变量数值。

（5）Stop Simulation（终止仿真）模块

当该模块输入信号为非零值时，终止系统仿真。

3.4.12 信号源模块库

Sources（信号源）模块库包含各种信号生成模块，如图 3-55 所示，主要分模块与子系统输入信号源和信号生成器两部分。模块与子系统输入信号源主要包括“In1”子系统输入接口模块，“Ground”信号接地模块，“From File”模块用于从数据文件输入数据，“From WorkSpace”用于从工作空间输入数据。信号生成器主要由 Step（阶跃信号）、Constant（恒值信号）、Sine Wave（正弦波信号）、Repeating Sequence（三角波信号）、Pulse Generator（脉冲信号）、Ramp（斜坡信号）、Clock 和 Digital Clock（时钟信号）等模块组成。

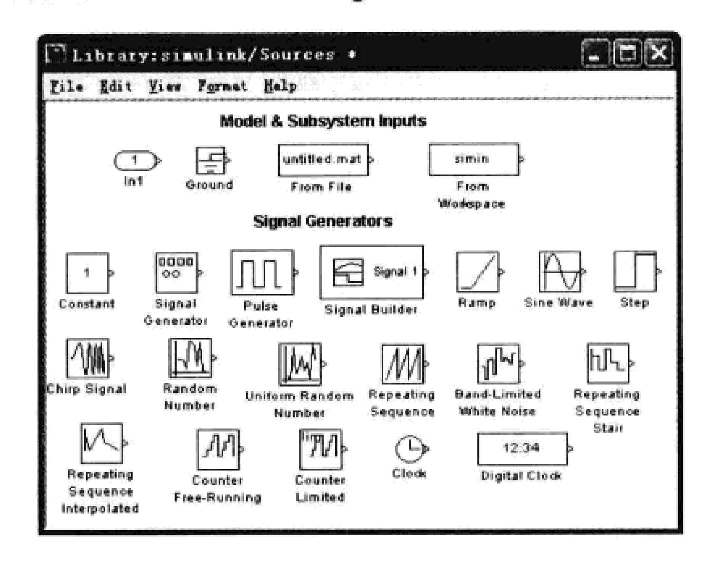


图 3-55　信号源模块库

3.4.13 用户自定义功能模块库

Simulink 提供了一个扩展功能模块库，如图 3-56 所示，主要包括以下内容。

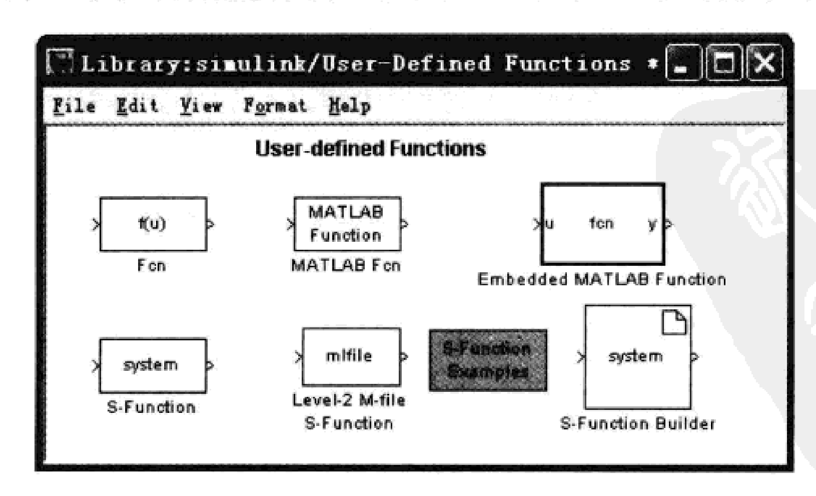


图 3-56　用户自定义模块库

（1）Fcn（函数功能）模块

这里可对输入变量进行各种数学操作。要注意的是，对于多输入变量，各变量的引用格式

为 $u(1)$、$u(2)$、$u(3)$等，而不是模块说明中的 $u[1]$、$u[2]$、$u[3]$。

（2）MATLAB Fun（MATLAB 功能）模块

这里可使用 MATLAB 规定的各种功能函数对输入变量进行操作，如 sin、cos 等。

（3）Embedded MATLAB function（嵌入式 MATLAB 功能函数）

双击该模块可打开带基本函数说明的 M 文件，如图 3-57 所示。在这里可对输入变量进行各种函数及数学操作。

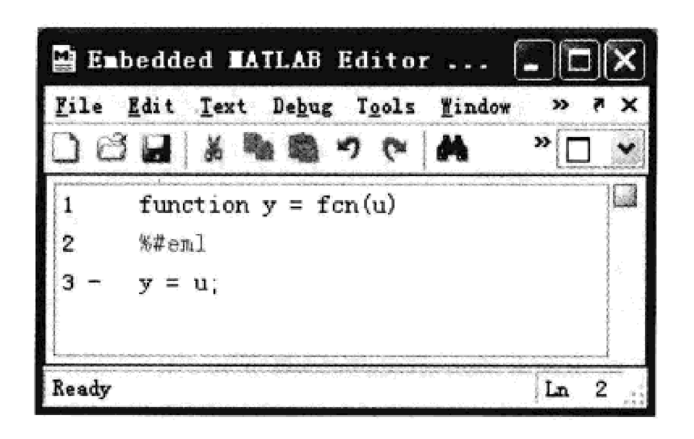


图 3-57 嵌入式 MATLAB Function 模块功能编辑窗口

（4）S-function（S-函数）模块

S-函数是系统函数（System Function）的简称，是 MATLAB 为用户提供的一个功能强大的接口。

（5）Level-2 S-函数

该模块适用于扩展的 S-函数文件，这里不多作介绍。

（6）S-函数编辑器模块

Simulink 提供了图形化的 S-函数编辑器，打开 S-函数编辑窗口，如图 3-58 所示。用户需要了解基本的 S-函数工作原理后才能方便地使用 S-函数编辑器。有关介绍将在第 4 章展开。

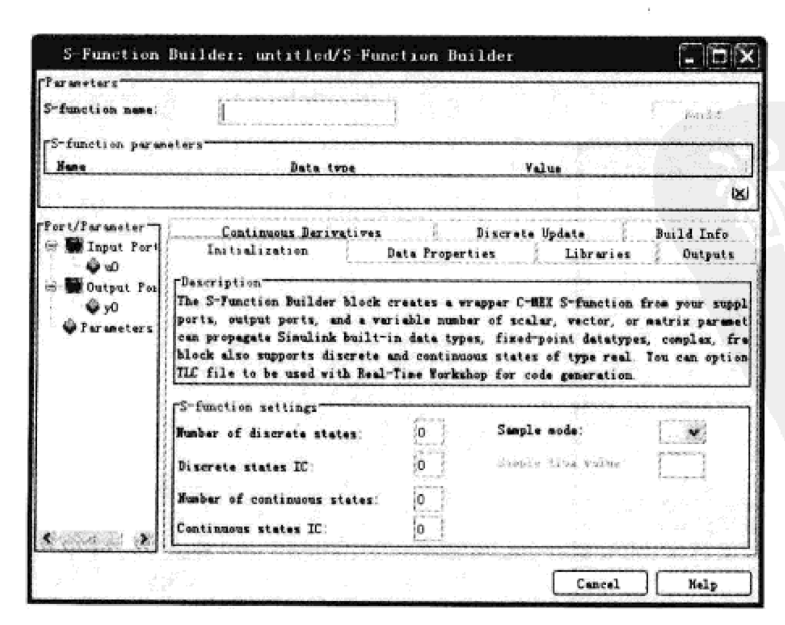


图 3-58 S-Function Builder 编辑窗口



（7）S-函数模板及实例源代码模块库

该模块库包括用 M 文件、C 语言、C++语言、Ada 语言、FORTRAN 语言编写的 S-函数模板源代码及示例。

3.5 Simulink 仿真示例

通过前面的内容，大家应该都了解并初步掌握了 Simulink 的使用方法。使用 Simulink 仿真的基本过程如下：

1）启动 Simulink 并打开模型编辑窗口。打开模型编辑窗口有以下两种方法：

- 单击 Simulink Libraray Browser 工具栏上的🗋快捷按钮，即可打开模型编辑窗口。
- 单击 Simulink Libraray Browser 的“File”菜单下的“New”选项中的“model”命令，即可打开模型编辑窗口。

2）将所需模块添加到模型中。

3）设置模块参数，并连接各个模块组成仿真模型。

4）设置系统仿真参数。

5）开始系统仿真。

6）观察仿真结果。

下面通过几个示例进行说明。

【例 3-4】 使用 Simulink 产生一个 5s 时出现的单位阶跃输入信号，并在示波器中显示出来。

其实现步骤如下：

1）单击 Simulink 的 Library 窗口中“File”菜单下“New”选项中的“model”命令，打开一个新的模型窗口。

2）分别从 Sources（信号源）库、Sinks（接收）模块库中，用鼠标把 Step（阶跃信号发生器）、Scope（示波器）这两个标准模块拖至新打开的模型窗口。

3）双击“Step”模块，打开其属性设置对话框，并将其中的“Step time”设置为“5”，其他参数为默认值，如图 3-59 所示。

4）按要求将 Step 与 Scope 模块连接起来，绘制成功后，模型如图 3-60 所示。

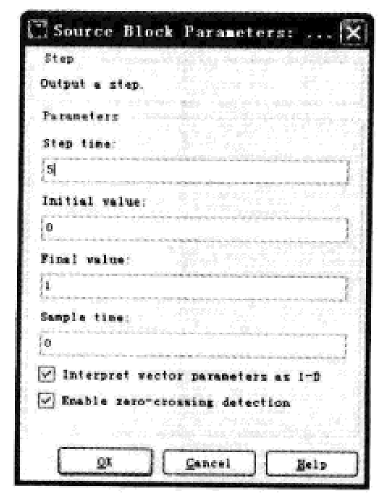


图 3-59 模块参数设置对话框

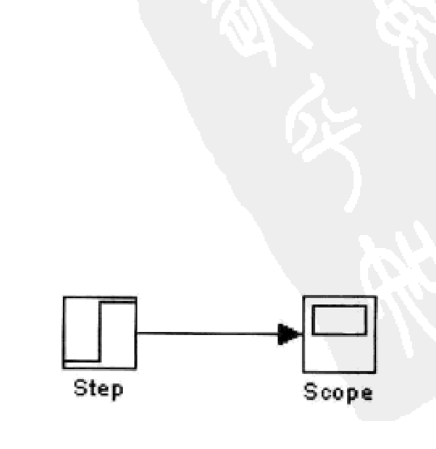


图 3-60 单位阶跃信号显示模型

5）单击▶按钮，对模型进行仿真。运行后，双击示波器，得到显示效果如图 3-61 所示。

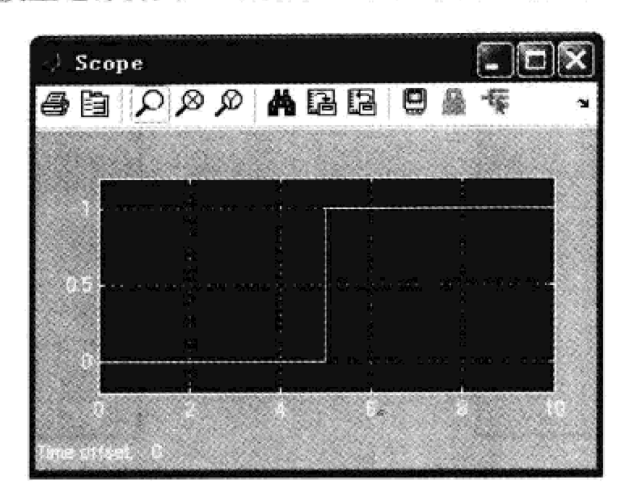


图 3-61　信号叠加的结果

【例 3-5】 产生一个 6sin(*t*)和和 sin(6*t*)叠加的信号，而且还叠加了功率谱为 1 的限带宽白噪声。

首先需要产生 6sin(*t*)、sin(6*t*)和限带宽白噪声信号，然后将这 3 个信号叠加起来。在此应用了正弦信号、限带宽白噪声、加法模块、示波器模块。分别将 Simulink Library Browser 中的以下模块依次拖动到 untitled 窗口中，模型效果图如 3-62 所示。

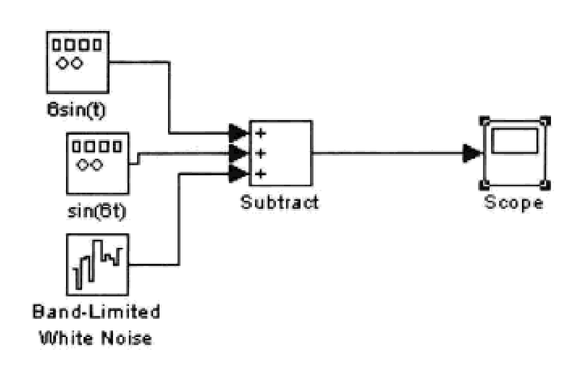


图 3-62　信号叠加模型

1）Sources 模块组中的 Signal Generator 模块，它是信号发生器模块。双击该模块，选定 sine 波形，设置其幅值为 6，频率为 1，这将产生 6sin(*t*)信号，设置参数窗口如图 3-63 所示。另一个模块，设置其幅值为 1，频率为 6，将产生 sin(6*t*)信号，设置参数窗口如图 3-64 所示。

2）Sources 模块组中的 Band-Limited White Noise 模块，它是限带宽白噪声模块。双击该模块，设置“Noise power”（功率谱）为 1，这将产生功率谱为 1 的限带宽白噪声信号。

3）Math Operations 模块组中的 Add 模块，它是加法模块，默认是两个输入相加，双击该模块，将“List of Signs”框中的两个加号（++）改为三个加号（+++），表明对 3 个输入量进行相加。

根据题意，设置好各模块参数并把各模块按顺序连接起来，运行仿真，其输出效果如图 3-65 所示。



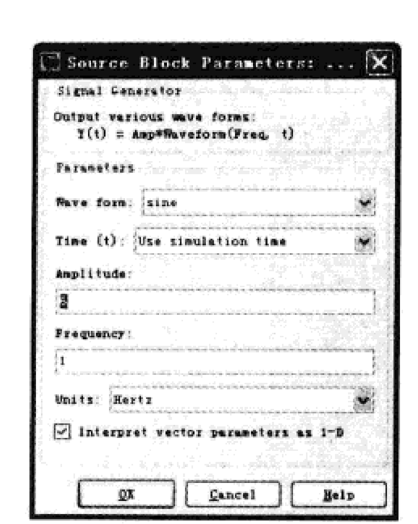


图 3-63　6sin(*t*)信号源参数设置

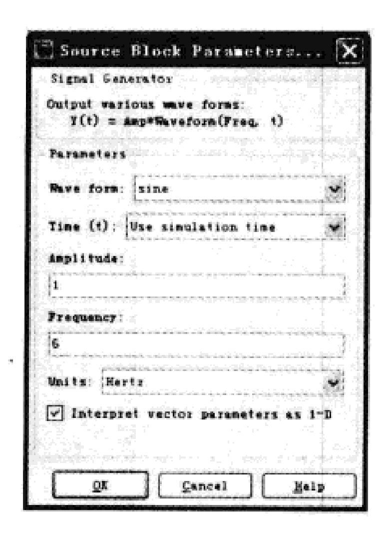


图 3-64　sin(6*t*)信号源参数设置

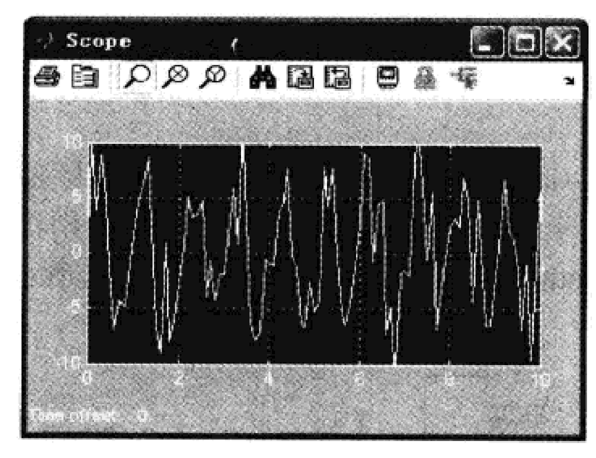


图 3-65　信号叠加显示效果



第 4 章　Simulink 建模与仿真高级应用



4.1　系统仿真建模的要求

利用 Simulink 仿真时，对于常见的、模型简单的仿真，对建模的方法性、有效性要求并不高；但随着模型的复杂，系统变得庞大，在采用 Simulink 建模仿真时，则需要进行科学、系统的规划。通常 Simulink 建模有以下基本要求：

（1）建模要从总体角度出发

这是指把一些个别的实体能成更大实体的程序，有时要尽量从能合并成一个大的实体的角度考虑对一个系统实体的分割。

（2）模型要有针对性

系统模型只应该包括与研究目的有关的方面，也就是与研究目的有关的系统行为子集的特征描述。对同一个系统，模型并不是唯一的，研究目的的不同，模型也不同。

（3）子系统划分要遵循一目了然规则

一个较大的系统往往由许多子系统组成，因此对应的系统模型也由许多子模型组成。在子模型与子模型之间，除了为实现研究目的所必需的信息联系之外，相互耦合要尽可能少，结构尽可能一目了然。

（4）模型精度要恰当

同一个系统的模型按其精确程序要求可以分为许多级。对不同的工程，精确程序要求也不一样。例如，用于飞行器系统研制全过程的工程仿真器要求模型的精度较高，甚至要考虑到一些小参数对系统的影响，这样复杂的系统模型，对仿真计算机的性能要求也高；但用于训练飞行员的飞行仿真器，对模型的精度要求则相对低一些，只要被培训人员感觉“真”即可。

4.2　Simulink 模块子系统

在 Simulink 仿真平台下，对于简单的动态系统，涉及元器件比较少，功能简单，那么可以直接使用 Simulink 建立模型进行仿真，但是对于大型的复杂系统，由于涉及模块较多，而且系统由若干个独立的功能模块构成，如果仅仅简单地使用 Simulink 基本模块库模块直接建立仿真模型，势必带来两个方面的问题：一方面模型看起来条理不太清晰，众多的仿真模块同时显示在仿真模型界面上，不便于模型的分析和修改，可移植性特别差；另一方面，系统模型的检测和调试将非常复杂。

针对一些庞大的仿真模型，通常将各个独立功能部分封装成子模块，先单独调试子模块，最后将各子模块综合在一起，进行系统模型的综合调试，这样整个系统条理清晰，系统

功能性特别强，而且由于各独立功能子模块进行了封装，可移植性大大加强。

4.2.1 子系统的生成与封装

明确子系统的基本含义以及在复杂仿真模型中的作用后，本节将通过演示子系统的创建和封装技术。

1. 创建子系统

在 Simulink 仿真平台下，创建子系统，通常有以下两种方法。

（1）在已经建立好的仿真模型中创建子系统

选择需要组合成子系统的相关模块，单击鼠标右键选择“Create Subsystem”选项，就可以组合成一个子系统，将所有相关模块封装到子系统中，输入/输出端口以 In1~Inx、Out1~Outx 形式表示。

图 4-1 是创建子系统前的仿真模型，图 4-2 是创建子系统后的仿真模型。此时没有对子系统进行任何操作，因此用户也无从得到子系统完成什么功能。需要查看子系统的仿真模型时，如果子系统没有被封装，那么直接双击子系统即可。当子系统被封装后，需要查看子系统仿真模型时，选中封装子系统，然后单击鼠标右键，在弹出的菜单中选择“Look under mask”命令即可。

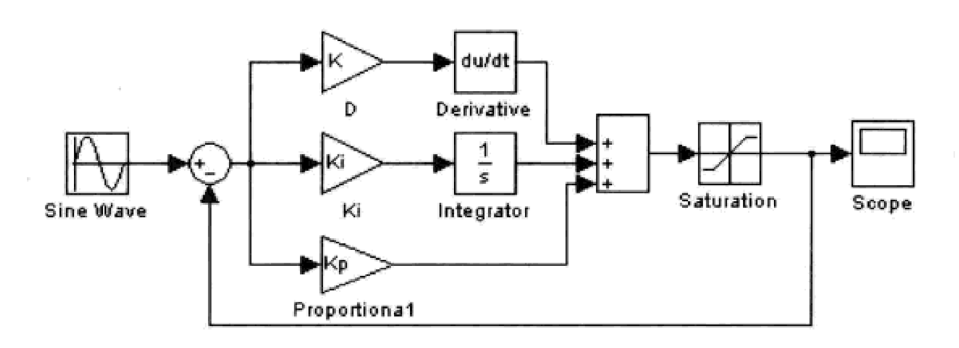


图 4-1 创建子系统前的仿真模型

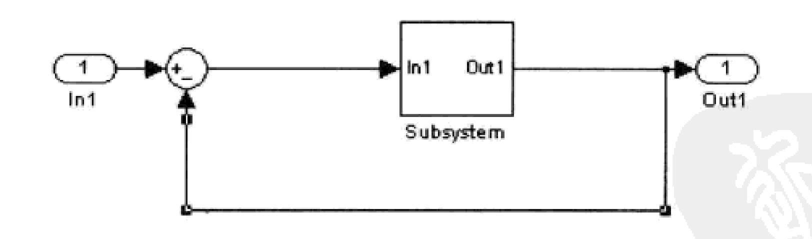


图 4-2 创建子系统

（2）在仿真模型中使用 Subsystem 模块建立空白子系统

选择 Subsystem 模块库中的 Subsystem 模块，双击该模块，即可以编辑子系统的模型。在空白的子系统中，只有一个输入端口和一个输出端口，对于多输入多输出子系统，可以添加输入/输出端口。

如图 4-3 所示，图 4-2 是建立系统仿真模型，然后双击“Subsystem”模块，按照图 4-3 所示更新子系统模块。可以看出，创建子系统模块的这两个过程实际上是一个相反的过程。创建子系统的方法有两种：方法一是先建立好子系统模型，然后创建子系统；方法二则是先

创建一个空白子系统，然后更新子系统模型。

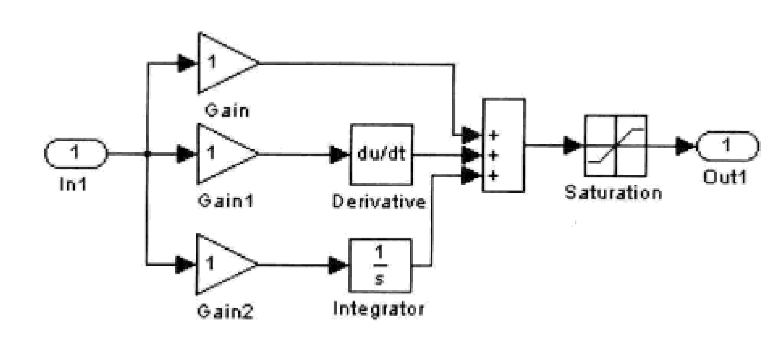


图 4-3 创建空白子系统

2．封装子系统

创建如图 4-2 所示的子系统后，需要对子系统进行封装，以便外部参数的输入。如果要封装子系统，选中创建的子系统，单击鼠标右键，在弹出的菜单中选择“Mask subsystem”选项，系统弹出如图 4-4 所示的封装子系统属性对话框。

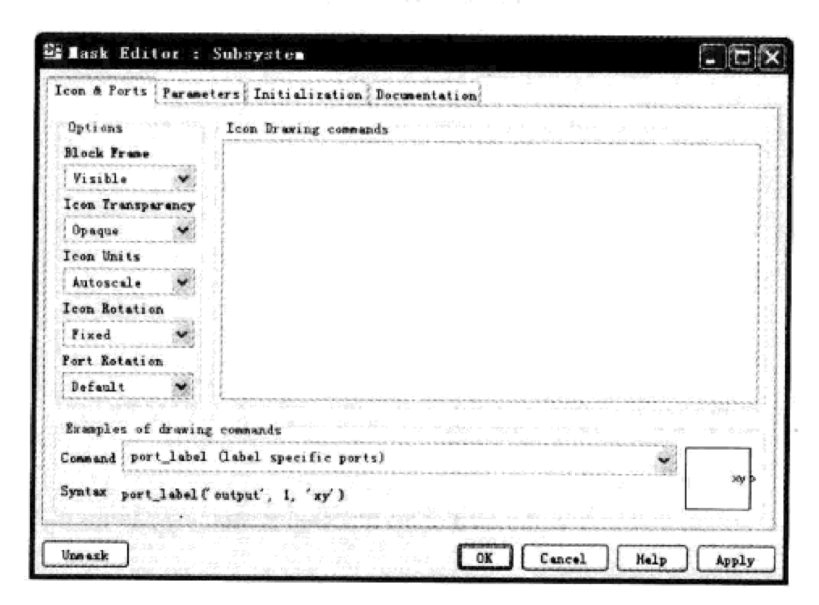


图 4-4 子系统封装图标设置属性对话框

（1）“Icon & Ports”选项卡设置

如图 4-4 所示，在“Icon & Ports”选项卡设置下，主要设置子系统封装图标。在“Icon Drawing commands”文本窗口中，可以使用子系统图标命令来个性化子系统图标。表 4-1 为所有相关的命令、功能描述，以及常用的调用格式。

表 4-1 创建子系统图标常用命令

函 数 名	功 能 描 述	调 用 格 式
disp	在封装模块中央显示 text 文本或者 Tex 模式文本	disp(text, 'texmode', 'on')
image	在封装模块中显示图片	image(a,[x,y,w, h],rotation)
sprintf	在封装模块中显示可变的 text 文本	fprintf(format, var)
dpoly	在封装模块中央显示传递函数形式，默认为“s”	dpoly(num,den,'character')
text	在封装模块的[x, y]位置显示 text 文本或者 Tex 文本	text(x, y, 'text',texmode,'on')

（续）

函 数 名	功 能 描 述	调 用 格 式
plot	在封装模块中绘制折线	plot(x1, y1, x2, y2,⋯)
patch	在封装模块中显示数据点块	patch(x, y, [r g b])
port_label	在封装模块的输入/输出端口旁绘制图标	port_label('port_type', port_number, 'label', 'texmode', on)

可以尝试在封装模块的“Icon Drawing commands”命令窗口中输入下列命令：

```
%以 Tex 模式显示方程α²+β²=γ²
disp('{\itEquation:} \alpha^2+\beta^2=\gamma^2','texmode','on')
%显示传递函数模型
dpoly([1 -1],[1 2 1],'p');
%显示零极点的离散模型
droots([1],[-1 2],5,'z');
%显示图片文件 matlab2009a_icon.bmp，此图片需要在当前路径下
image(imread('matlab2009a_icon.bmp'))
%在子系统输入输出端口显示图标
disp('Car\nSwaper');
port_label('input',1,'\spadesuit','texmode','on');
port_label('output',1,'\heartsuit','texmode','on');
%显示散点块
patch([0 0.5 1],[0 1 0],[1 0 0]);
%绘制折线
plot([0 1 5],[0 0 4]);
%以 Tex 模式显示文本
text(0.05,0.5,'{\itEquation:} \Sigma \alpha^2+ \beta^2 \rightarrow \infty, \pi,...
 \phi_3={\bfcool}','hor','left','texmode','on');
```

（2）“Parameters”选项卡设置

“Parameters”选项卡用来封装子系统模型中的变量参数，如图 4-5 所示。

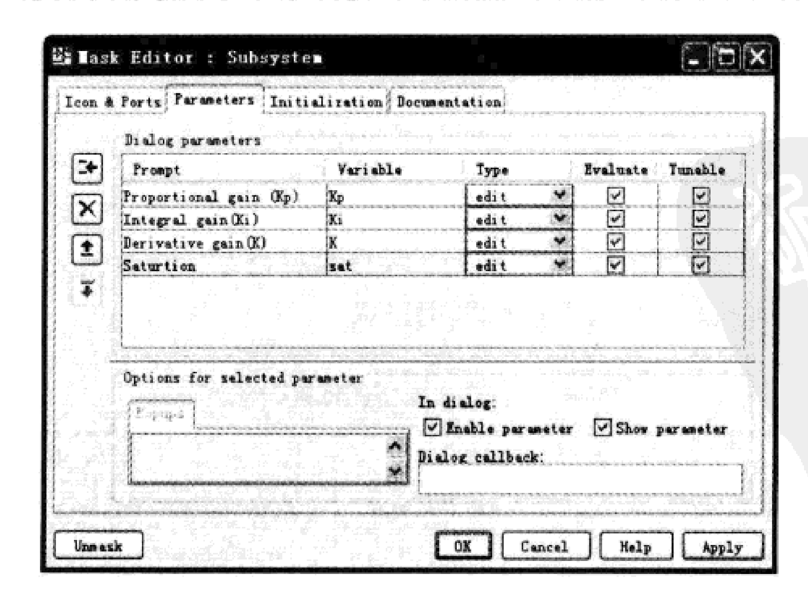


图 4-5 “Parameters”选项卡设置

在“Parameters”选项卡设置中，最左边的 4 个按钮从上到下功能分别为：“添加变量”、

"删除变量"、"移到上一变量"和"移到下一变量"。

（3）"Initialization"选项卡设置

"Initialization"（初始化）选项卡设置如图 4-6 所示。在"Initialization"选项卡设置中，可以设置子系统模块的初始化信息，包括变量的初始值设定、其他参数的相关运算等。



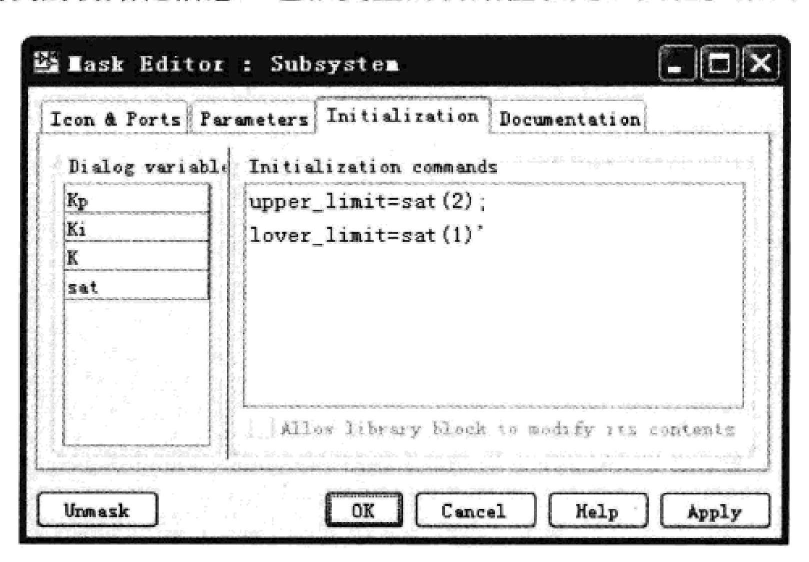


图 4-6 "Initialization"选项卡设置

（4）"Documentation"选项卡设置

"Documentation"（文档）选项卡用于设置子系统的封装类型（Mask type），即封装子系统的名称；封装描述，即对封装子系统的相关描述和封装的 Help 文档，其页面效果如图 4-7 所示。

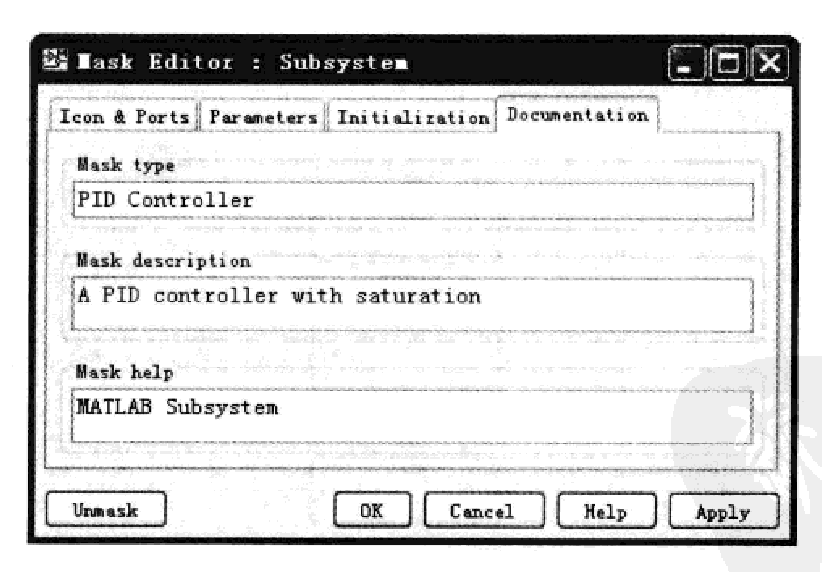


图 4-7 "Documentation"选项卡设置

按照上述步骤，就完成了子系统的封装，结果如图 4-8 所示。

4.2.2 触发子系统

在 Simulink 仿真开发平台下，除了用户自定义子系统模块外，还提供了另外一类子系统模块，即条件执行制系统模块，包括触发子系统、使能子系统、触发使能子系统、if/else 子系统、switch/case 子系统、while 子系统以及 for 子系统等。

这类子系统有一个共同的特点，即子系统的执行与否与外部的驱动信号直接相关，如触发子系统，必须在触发沿（上升沿、下降沿或者两者）到来时，子系统才被执行，使能子系统同样需要外部正电平的驱动才能执行子系统。

Function Block Parameters: Subsystem

PID Controller (mask)

A PID controller with saturation

Parameters

Proportional gain (Kp)

2.5

Integral gain(Ki)

5.2

Derivative gain(K)

0.05

Saturtion

[-10 10]

OK　Cancel　Help　Apply

图 4-8　子系统封装结果

如图 4-9 所示的触发子系统仿真模型的系统工作原理：脉冲发生器产生脉冲信号，也即产生上升沿和下降沿，当产生上升沿时（信号由 0 变为 1），驱动第一个子系统；当产生下降沿时（信号由 1 变为 0），驱动第二个子系统；当产生上升沿或者下降沿时，均可以驱动第三个子系统。

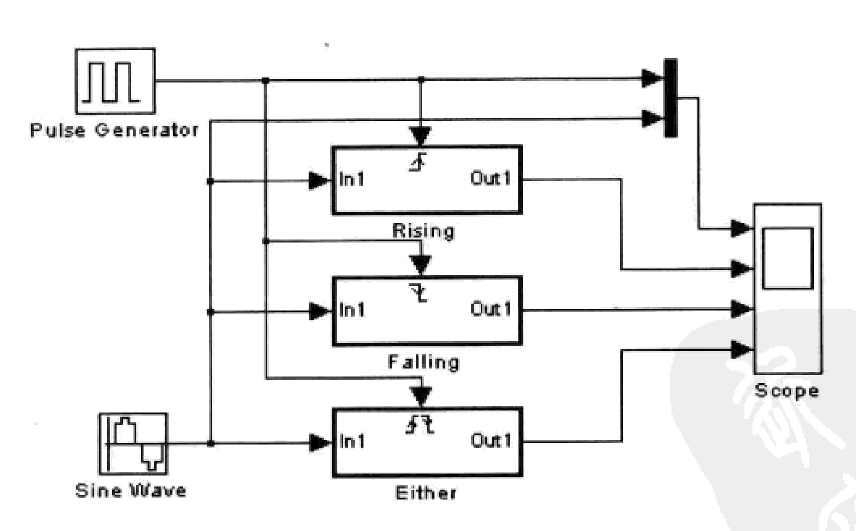
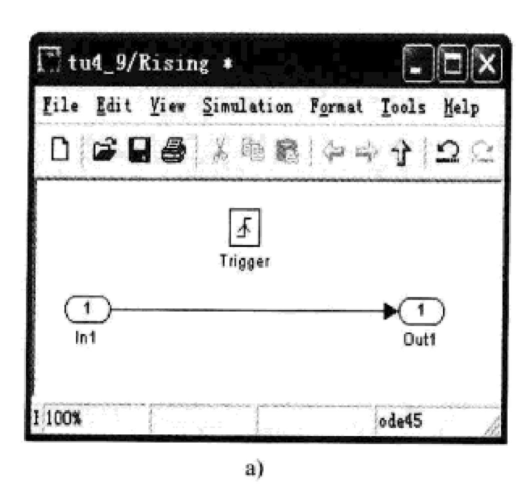


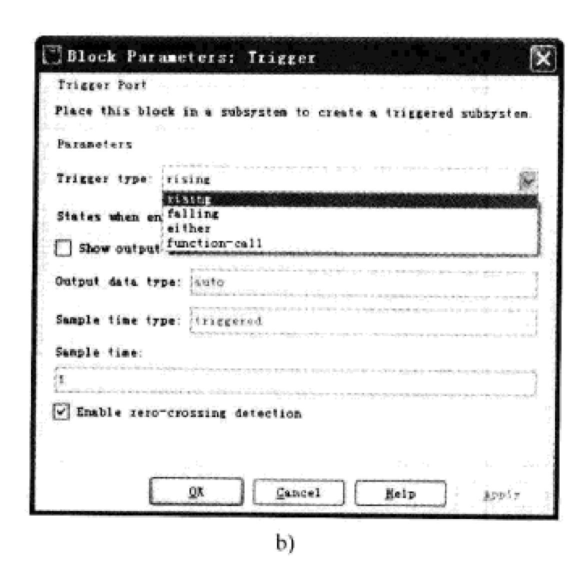
图 4-9　触发子系统仿真模型

脉冲发生器的参数设置如下：幅值为 1，周期为 1s，占空比为 50%。正弦波信号的参数设置如下：基于时间采样，幅值为 1，周期为 8s，初始相位为 0，采样时间为 1ms。

图 4-10 所示为触发子系统及属性设置，在触发类型的下拉列表框中选择上升沿触发（Rising）、下降沿触发（Falling）或两者之一（Either）。触发类型（Trigger type）下的灰色按钮表示触发子系统后系统输出保持，这个选项在触发子系统中无法编辑。仿真结果如图 4-11 所示。



a)



b)



图 4-10 触发子系统及属性设置



a) 触发子系统图 b) 子系统属性设置

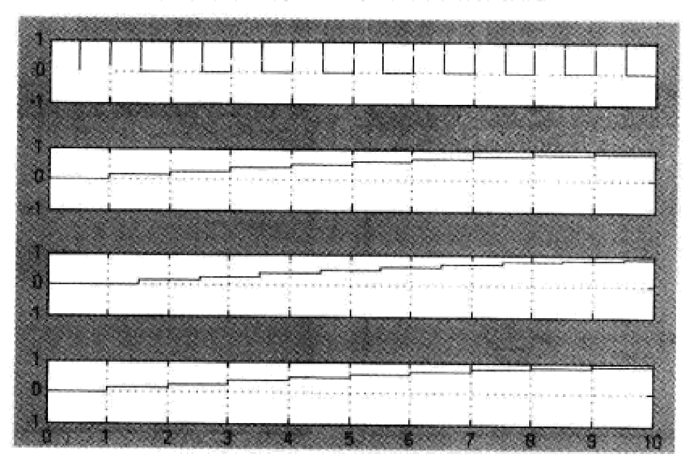


图 4-11 触发子系统仿真结果

4.2.3 使能子系统

如图 4-12 所示的使能子系统的工作原理描述如下：脉冲发生器产生脉冲信号经过布尔逻辑，转化为逻辑控制信号，控制两个使能子系统。当方波逻辑为 1 时，执行上一个使能子系统，当方波为-1 时，经过布尔逻辑反操作（Not）模块，从而控制下一个使能子系统工作。偏移常数[0 2]和[2 0]是为了方便观察两个输入信号之间差异。

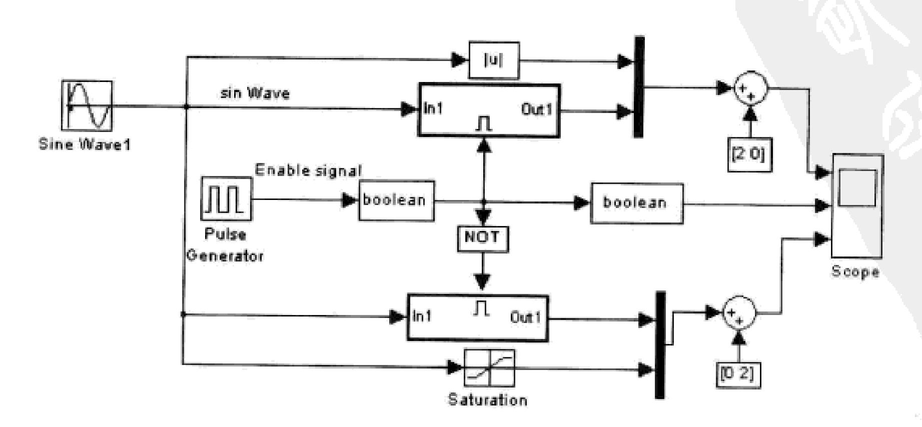
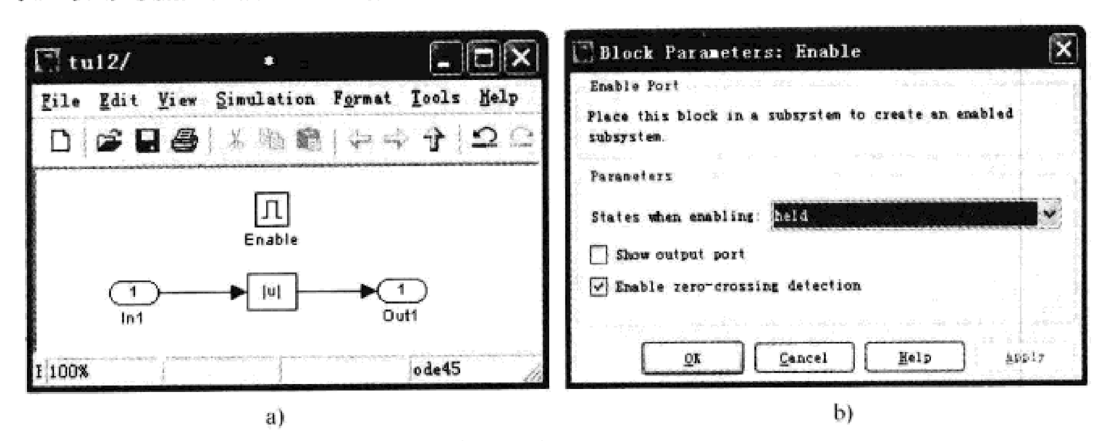


图 4-12 使能子系统仿真模型

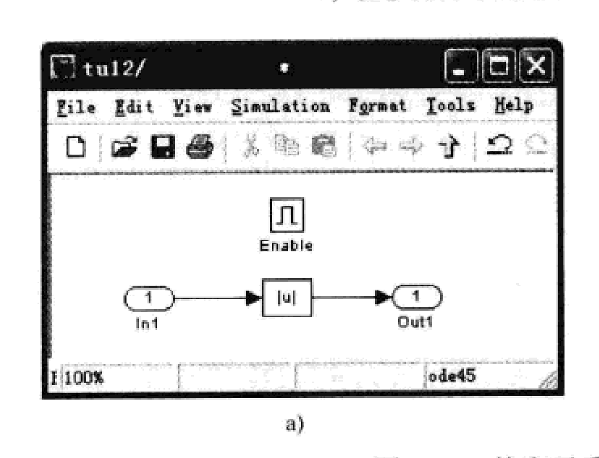
如图 4-13 和图 4-14 所示分别为取绝对值子系统和饱和子系统的使能设置。当选择使能状态为“reset”时，表示使能子系统开始执行时，系统中的状态被复位；当选择使能状态为“held”时，表示使能子系统开始执行时，系统中的输出状态保持不变。

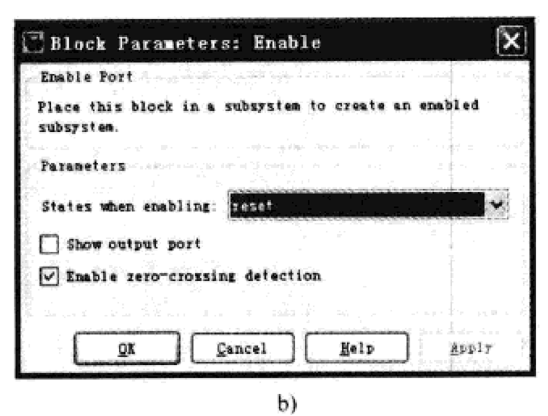


a)　　　　　　　　　　b)

图 4-13　取绝对值系统使能及其设置

a) 取绝对值子系统模型　b) 取绝对值系统使能设置





a)　　　　　　　　　　b)

图 4-14　饱和子系统使能及其设置

a) 饱和子系统模型　b) 饱和子系统使能设置

使能子系统仿真结果如图 4-15 所示。

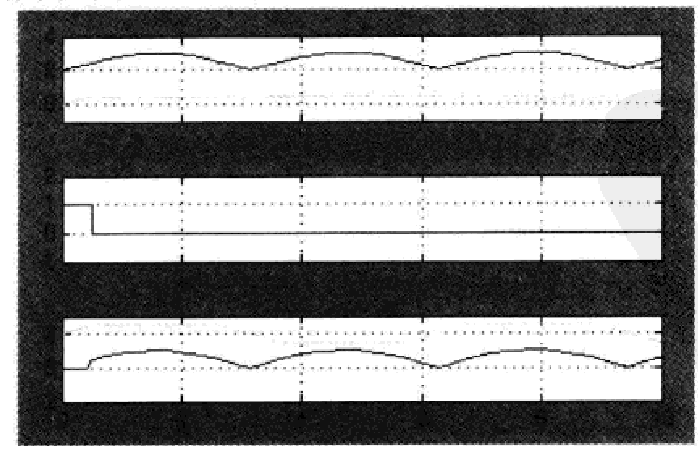


图 4-15　使能子系统仿真结果

4.2.4 触发使能子系统

触发使能子系统的执行条件同时包含了触发和使能，触发使能子系统的执行受到触发信

号和使能信号的共同控制，只有两个条件都满足时，子系统才会执行。

触发使能子系统的工作原理如下：如果子系统的使能信号满足条件，那么当触发信号到来时，子系统被执行，同样如果存在触发信号，需要等待使能信号端为正电平时，子系统才能被执行。只有触发信号和使能信号同时满足条件时，子系统才能工作。



如图 4-16 所示的触发使能子系统工作原理如下：两个脉冲发生器分别产生触发信号和使能信号，使能信号的周期是触发信号周期的两倍，触发使能子系统 A 在使能信号为正电平，且触发信号为上升沿时开始工作。触发使能子系统 B 在使能信号为正电平且触发信号为下降沿时开始工作。触发使能子系统 C 在使能信号为正电平且触发信号为上升沿或下降沿时开始工作。

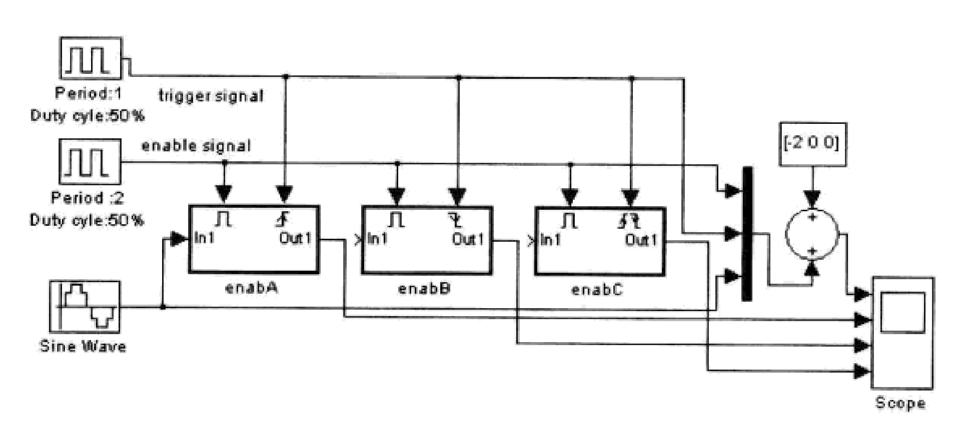


图 4-16 触发使能子系统仿真模型

脉冲发生器参数设置如图 4-17 所示。正弦波发生器参数设置如下：基于时间采样，幅值为 1，周期为 6s，相位为 0，采样周期为 1ms。触发使能子系统 A、B、C 的参数设置，如图 4-18 所示。子系统模型中，显示了各子系统使能的状态以及输出端口在不使能条件下的输出状态和初始值。

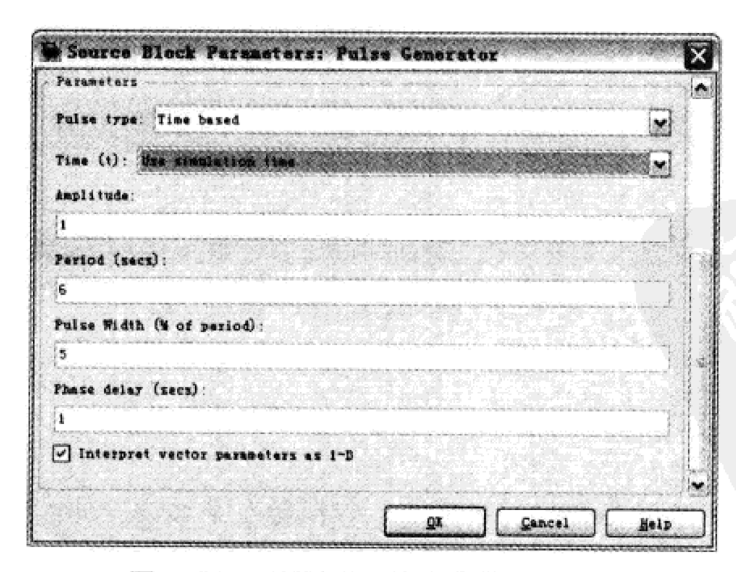
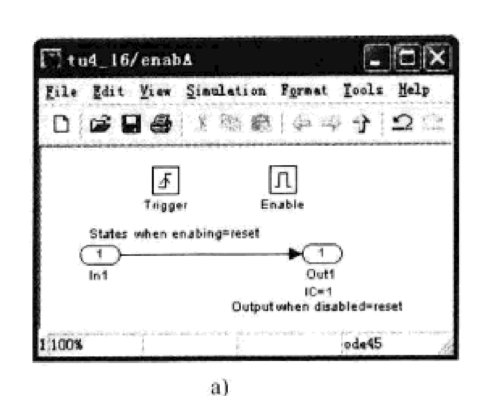
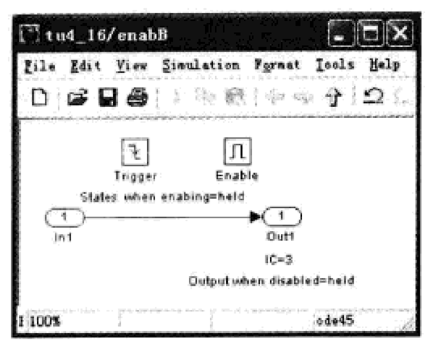


图 4-17 正弦波发生器参数设置对话框

触发使能子系统 A 当使能时复位，而输出端口不使能时复位，初始值为 3；触发使能子系统 B 使能状态保持，而输出端口不使能时状态保持，初始值为 1；触发使能子系统 C 使能状态复位，而输出端口不使能时复位，初始值为 5。仿真模型结果如图 4-19 所示。



a)



b)

c)

图 4-18　触发使能子系统设置

a) 上升沿　b) 下降沿　c) 两边沿

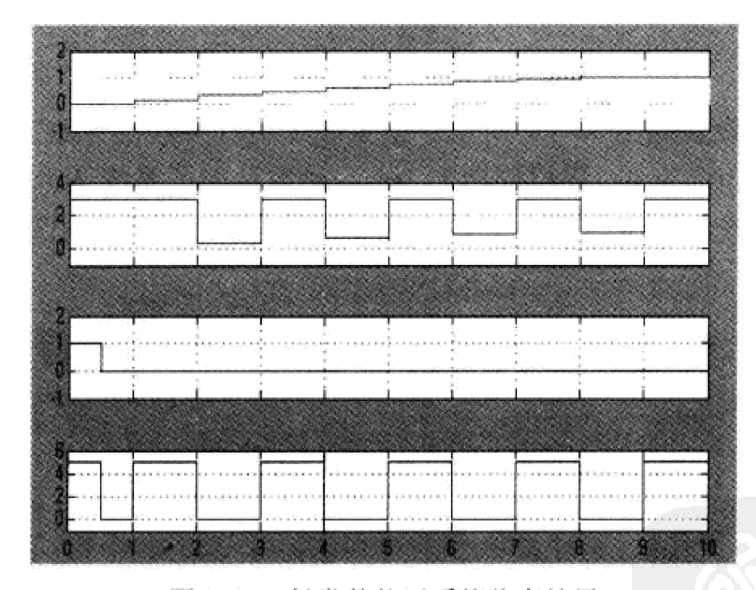


图 4-19　触发使能子系统仿真结果

4.2.5 受控子系统

1. if/else 子系统

在前面已经介绍了 3 种条件执行子系统，包括触发子系统、使能子系统和触发使能子系统。通过条件判断语句 if/else、while 或者 for 等循环语句同样可以实现条件执行子系统。

if/else 子系统工作原理可以描述为：判断 if/else 的条件指令，如果 if 逻辑为真，则执行 if 子系统，否则 else 逻辑为真，执行 else 子系统。if/else 子系统可以独立为 if 子系统。对于如图 4-12 给出的使能子系统仿真模型，同样可以用 if/else 子系统来实现，if/else 子系统仿真模型如图 4-20 所示。



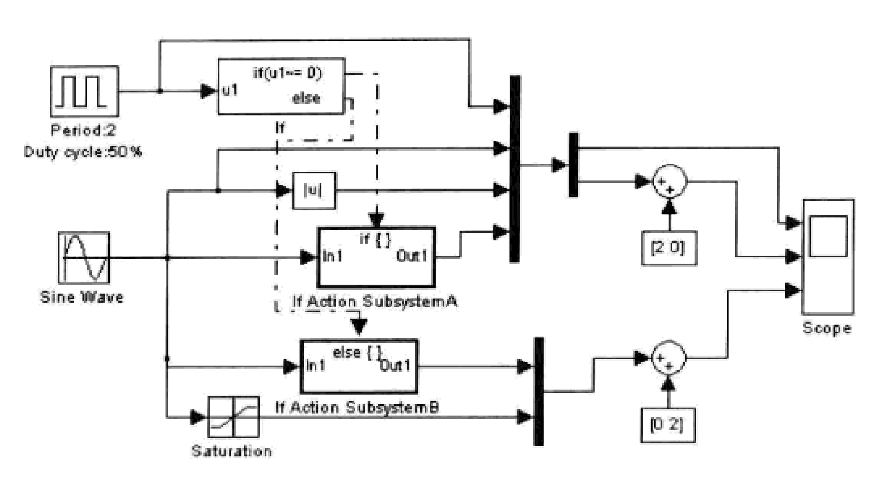



图 4-20　if/else 子系统仿真模型

仿真模型中，脉冲发生器的参数设置为：基于时间采样，周期为 2s，占空比为 50%。正弦波信号发生器参数设置为：基于时间采样，幅值为 1，周期为 2s，相位为 0。if 模块参数设置如图 4-21 所示，模型仿真如图 4-22 所示。

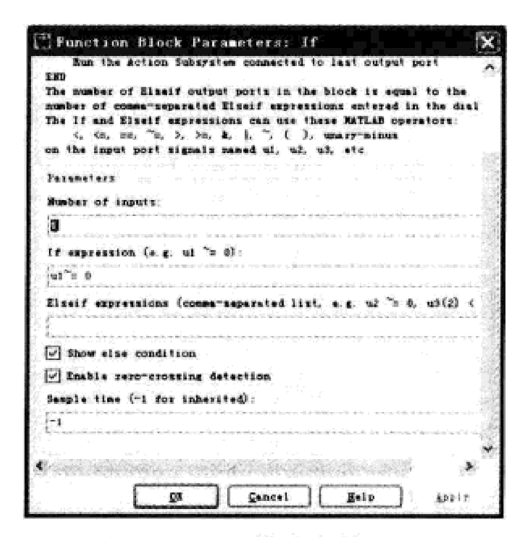
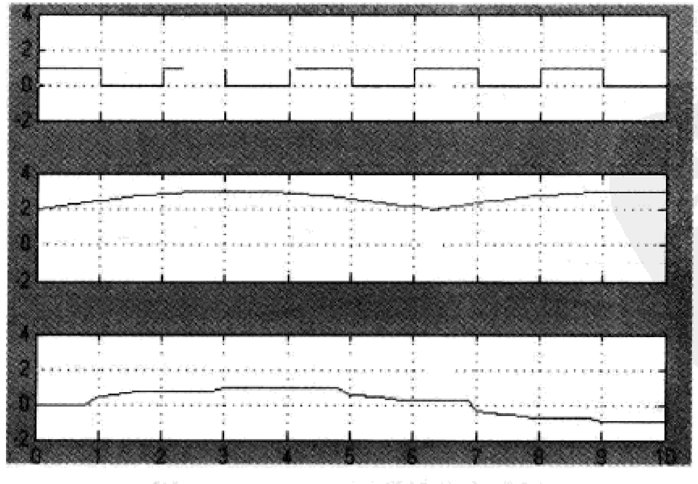


图 4-21　if 模块参数设置


图 4-22　if/else 子系统仿真结果

2．switch/case 子系统

switch/case 子系统可以实现对于不同输入情况根据 case 语句分别执行不同的 case 子系统。对于如图 4-20 示的 if/else 子系统仿真模型，同样可以用 switch/case 子系统来完成。如

图 4-23 所示为 switch/case 子系统仿真模型，工作原理可以描述为：利用脉冲发生器产生 0.1 的脉冲信号，如果脉冲信号为 1，那么执行 case[1]对应的子系统，如果脉冲信号为 0，则执行 case[0]所对应的子系统。

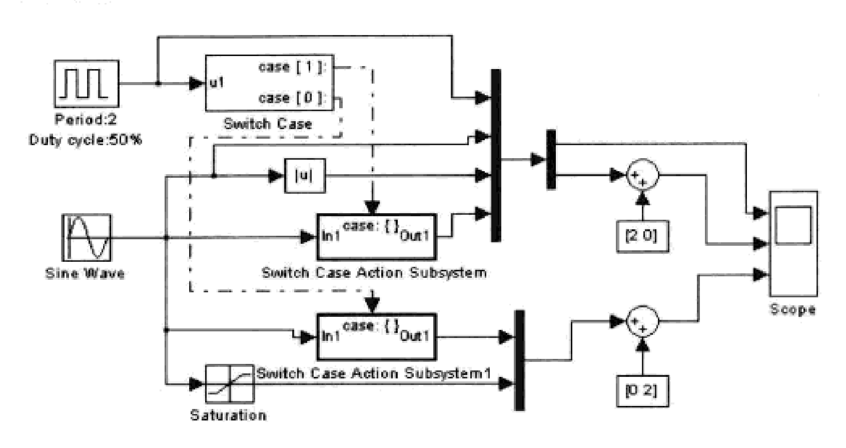


图 4-23　switch/case 子系统仿真模型

switch/case 子系统属性设置如图 4-24 所示，仿真结果如图 4-25 所示。

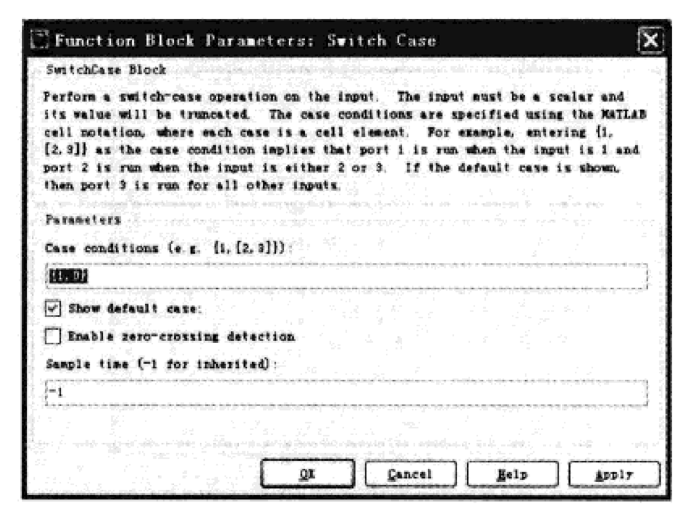


图 4-24　switch/case 子系统属性设置

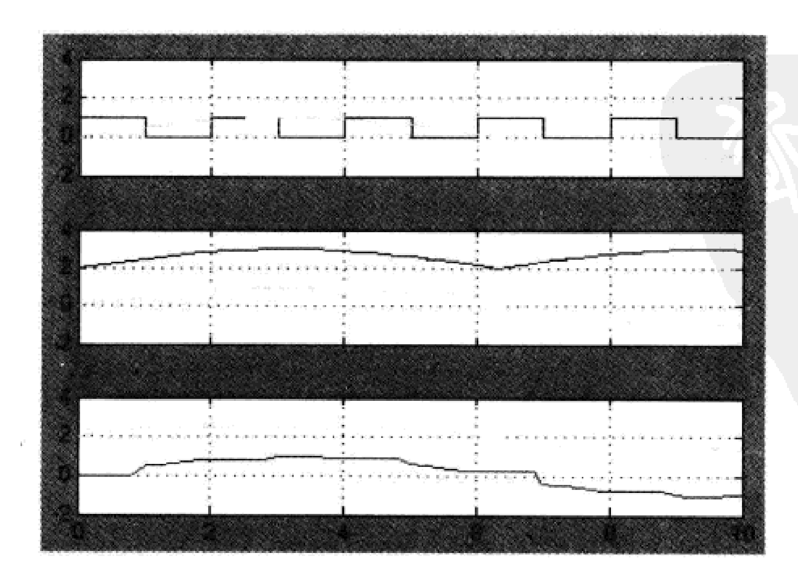


图 4-25　switch/case 子系统模型仿真结果

3. while 子系统

while 子系统可以实现在一个时间步长内，如果 while 条件为真的情况下，重复执行子系统。它的功能和 C 语言中的 while 语句完全相同。在 Simulink 仿真平台下，通过选择 while 循环类型可以实现两种循环执行子系统：while 子系统和 do-while 系统。对于 do-while 子系统，只有一个条件输入，而且必须位于 while 子系统的内部，在每个时间步长下，while 子系统检测条件输入的逻辑是否为真。如果为真，则重复执行子系统，只有当条件输入逻辑为假或者迭代次数达到最大的迭代次数时，退出 while 子系统。而 while 子系统包含两个输入：一个为条件输入；另一个为初始值输入。初始值输入必须在 while 子系统的外部，在每个时间步长上，while 子系统首先检测初始值输入逻辑，如果为真，则进行 while 子系统，同时检测条件输入逻辑是否为真，如果是真，则重复执行子系统，只有当条件输入逻辑为假或者迭代次数达到最大的迭代次数时，退出 while 子系统。如图 4-26 所示为 while 与 do-while 子系统仿真模型，分别使用了 do-while 循环类型和 while 循环。do-while 子系统和 while 子系统模型及属性参数设置如分别如图 4-27 和图 4-28 所示。



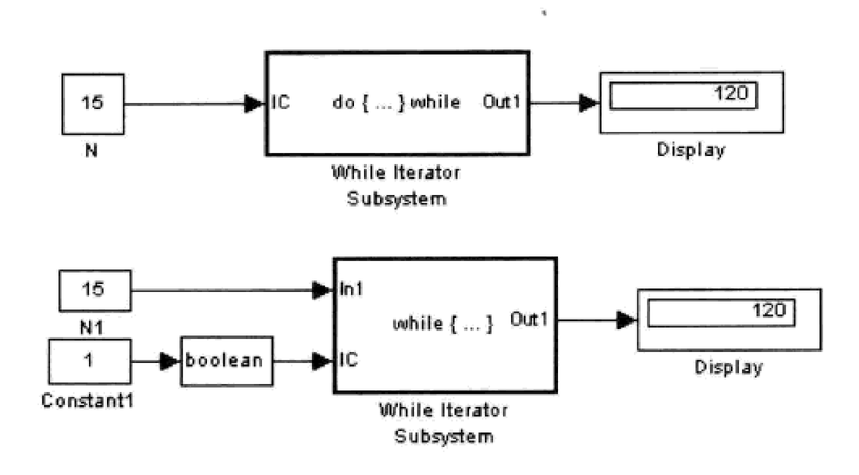
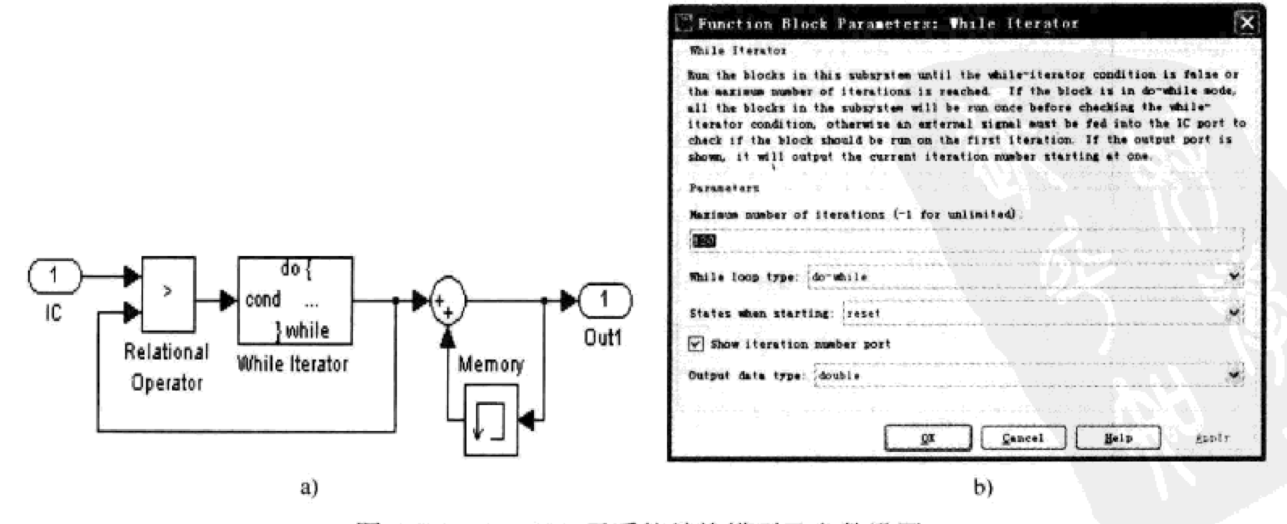


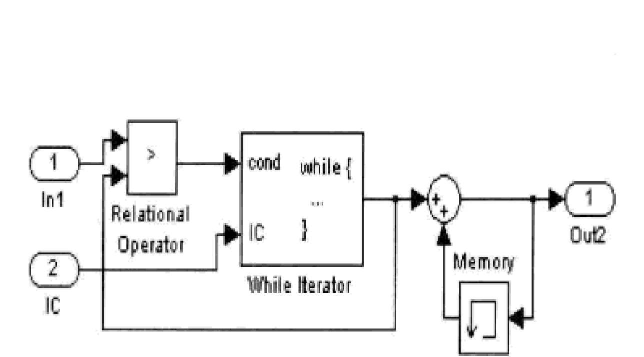
图 4-26 while 与 do-while 子系统仿真模型



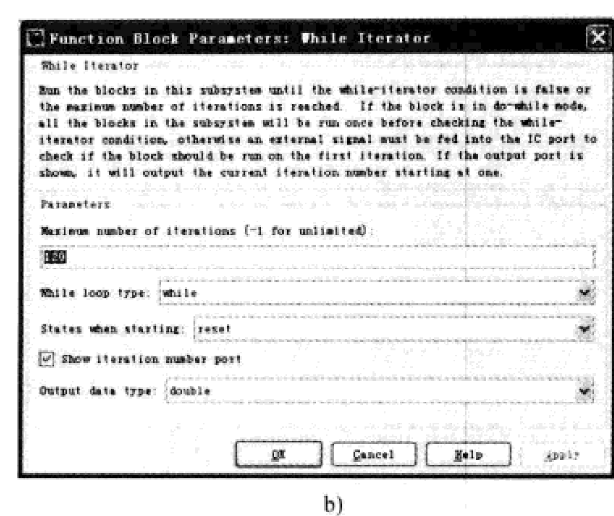
a) b)

图 4-27 do-while 子系统结构模型及参数设置

a) do-while 子系统结构模型 b) do-while 子系统参数设置



a)



b)

图 4-28 while 子系统结构模型及参数设置

a) while 子系统结构模型 b) while 子系统参数设置

在 while 子系统仿真模型运行开始后，while 子系统自动设置迭代起始值为 1，于是如图 4-26 所示的 while 子系统仿真模型实现了 1~15 的累加。用 MATLAB 语言可以描述如图 4-27 所示的仿真模型为：

```
>> sum=0;
i=0;
Muxstep=15;
while(i<Muxstep)
    i=i+1;
    sum=sum+i;
end
sum
```

注意：通常情况下，除了用户能够肯定条件输入会出现假逻辑时，一般应该给 while 子系统设置最大的迭代次数，以避免 while 子系统陷入列循环。

4. for 子系统

for 子系统和 while 子系统类似，实现系统的循环调用，直到 while 条件或 for 条件不满足后，退出子系统循环。如图 4-29 所示为 for 子系统实现 1~15 的累加仿真模型。图 4-30 为 for 子系统参数设置。

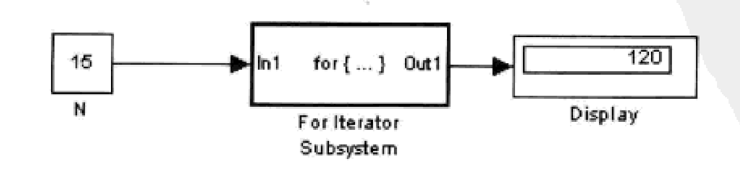


图 4-29 for 子系统仿真模型

注意：在例中如果实现 1~15 的累加，那么必须设置循环开始状态复位，否则，选择循环开始状态保持，则对于每一个 i，得到的加和结果被保持，相当于每次累加的结果再进行

累加，将无法得到正确的结果。

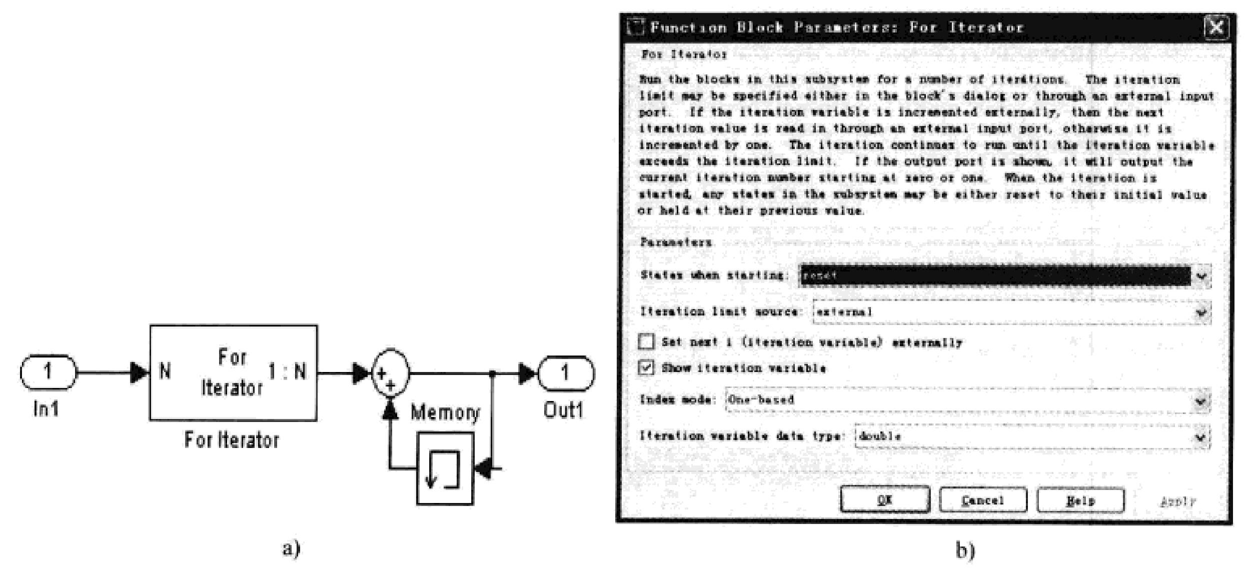


图 4-30 for 子系统结构模型及其参数设置

a) for 子系统结构模型 b) for 子系统参数设置



4.3 S-函数建模与仿真

4.3.1 S-函数介绍

1. S-函数简介

S-函数是 System Function（系统函数）的简称，是指用指定语言描述的一个非图形化功能模块（以区别 Simulink 的系统模块），是 MATLAB 为用户提供的一个扩展功能接口。用户可以采用 MATLAB 语言、C、C++、FORTRAN 或者 Ada 等语言编写 S-函数。S-函数由一种特定的语法构成，用来描述并实现连续系统、离散系统、混合系统等模型。S-函数可接收来自 Simulink 求解器的相关信息，并对求解器发出的命令作出响应，这种交互作用类似 Simulink 系统模块与求解器的交互作用。

S-函数作为 MATLAB 与其他语言相结合的接口，用户可以使用该语言所提供的所有强大功能。例如，MATLAB 语言编写的 S-函数可以充分利用 MATLAB 所提供的各种功能函数，使用 C 语言编写的 S-函数则可以实现对操作系统的访问。

虽然 Simulink 提供了大量内置模块并允许用户自定义功能模块，但是并不能完全满足用户的需要。特别是当开发一个新的通用的模块作为一个独立的功能单元时，使用 S-函数实现则是一种相当简便的方法。此外，还可以借助 S-函数使用其他语言编写的已有代码，实现某种程度上的代码移值。

总结起来，S-函数有如下特点：

- S-函数是 Simulink 的系统函数。
- 能够响应 Simulink 求解器的命令。
- 采样语言命令实现一个动态系统。
- 可以开发新的 Simulink 模块。

- 可以与已有的代码相结合进行仿真。
- 可以采用文本编辑方式输入复杂系统的数学模型。
- 扩展 Simulink 功能。
- S-函数的语法结构是为实现一个动态系统而设计的，其他 S-函数的用法是默认的特例。

2. S-函数的表示法

在动态系统设计、仿真与分析中，用户可以使用 Simulink 浏览器里用户自定义模块库中的 S-Function 模块来调用已编写的 S-函数。S-Function 模块默认为是一个单输入单输出的系统模块，如果有多个输入或者输出信号，可以采用 Mux 模块和 Demux 模块对信号进行组合和分离操作。在 Simulink 空白模块里粘贴 S-Function 模块，双击该模块弹出参数设置对话框，如图 4-31 所示。

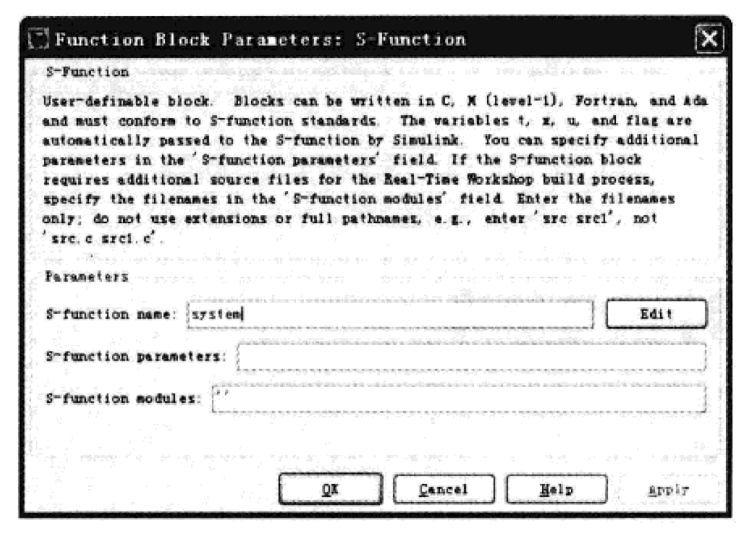


图 4-31　S-Function 参数设置对话框

在“S-Function name”文本框中输入已编写的 S-函数文件名，输入完成后单击“Edit”按钮，弹出该 S-函数文件编辑窗口。在“S-Function parameters”文本框中输入用户添加的参数，“S-Function parameters”文本框适用于用 C MEX-File 编写的 S-函数。

由上面对 S-函数的调用可知，Simulink 下的 S-Function 模块仅仅是用图形的方式完成调用 S-函数的一个接口。实际的功能需要由 S-函数的源文件来定义。一般而言，使用 S-函数的基本步骤如下：

- 创建 S-函数源文件。创建 S-函数源文件的方法有很多，用户可以按照 S-函数语法格式自行编写每一行代码，但是这样做既费时费力，又容易出错。Simulink 为用户提供了很多 S-函数的模板和例子，我们可以根据自己的需要修改相应的模板或者例子即可。详细的 Simulink 模板将在后面作进一步介绍。
- 在 Simulink 模型框图中添加 S-Function 模块，并进行正确的参数设置。
- 在 Simulink 模块框图中按照定义好的功能连接输入/输出端口。
- 进行基本仿真参数设置，单击工具栏中的“Run”快捷按钮，运行仿真。

注意：在 S-Function 模块中的 S-函数名称、S-函数参数列表必须和用户建立的 S-函数源文件的名称、函数参数名称、顺序完全一致，并且函数参数之间必须用逗号分开。此外，用户也可以使用子系统封装技术对 S-函数进行封装，以增强系统模型的可读性。



3. S-函数模板

为了方便用户编写和使用 S-函数，Simulink 提供了丰富的 S-函数的模板和示例。这里以寻找 M 文件编写的 S-函数模板为例，来说明 S-函数模板文件及示例的基本查找方法。

单击 Simulink Library 的“User-Definded Function”前的扩展按钮，再双击“S-Function Examples”模块，系统弹出如图 4-32 所示的 S-函数示例模块库，图中包括用 M 文件、C 语言、C++、Ada 和 FORTRAN 语言编写的 S-函数示例。

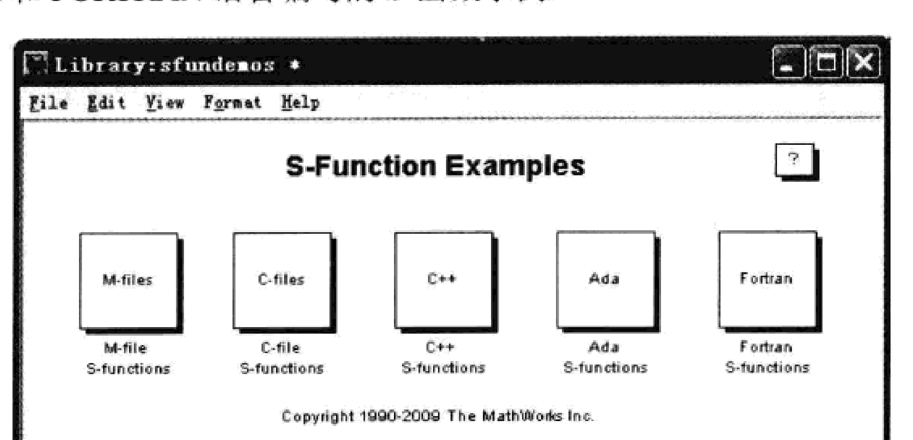


图 4-32 S-函数示例模块库

在图 4-32 中双击“M-file S-functions”模块，系统弹出如图 4-33 所示的用 M 文件编写的 S-函数模块库，它由“Level-1 M-files”和“Level-2 M-files”两部分组成。Level-1 M-file 子模块库中的 S-函数用于兼容以前版本的 S-函数仿真，Level-2 M-file 用于扩展 M 文件的 S-函数仿真。

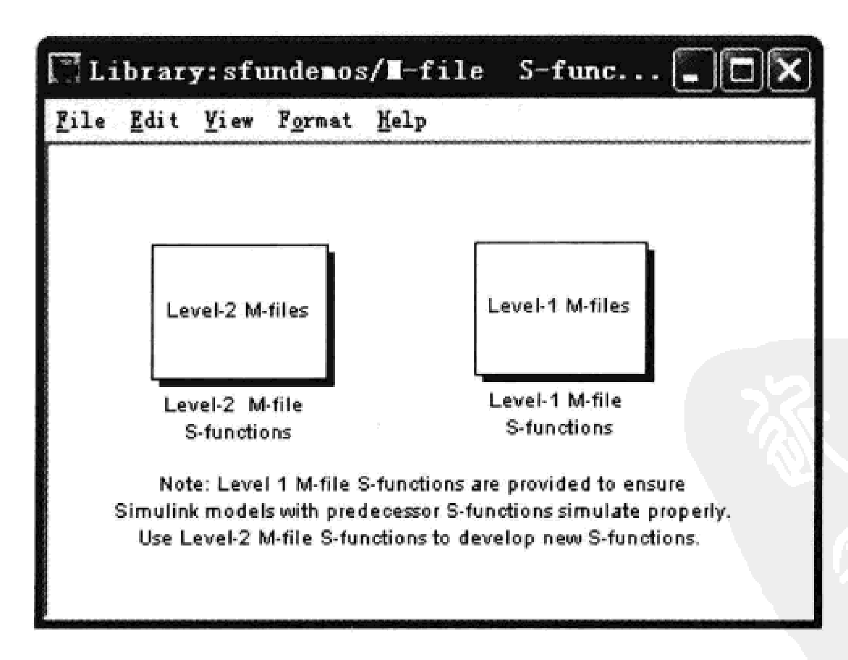


图 4-33 M 文件的 S-函数模块库

在图 4-33 中双击“Level-1 M-file”模块，系统弹出用 M 文件编写的 S-function 模块及示例的子模块库，如图 4-34 所示。下面以 Simulink 仿真示例来说明其基本用法。

【例 4-1】 以 S-函数完成对输入信号的运算：$y = 5u - 3$。

1）其实现的 Simulink 模型如图 4-35 所示。

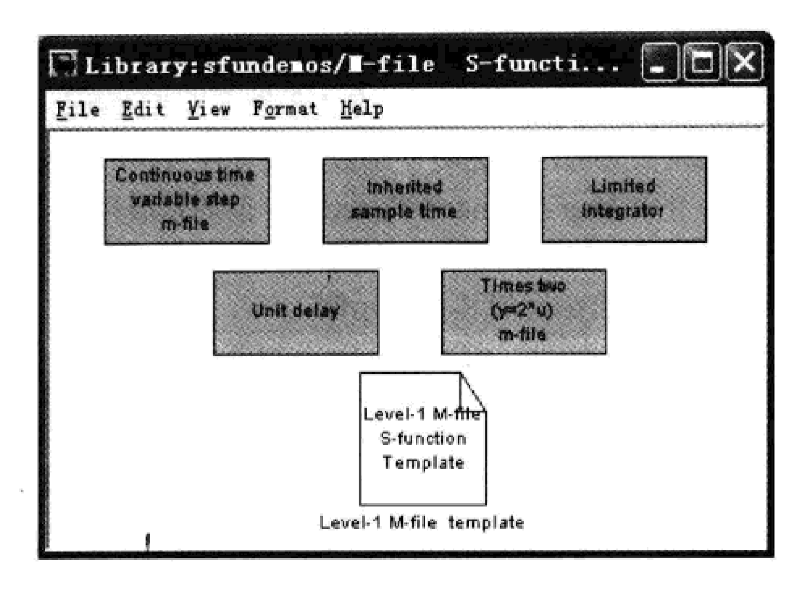


图 4-34　Level-1 M-file 模块库

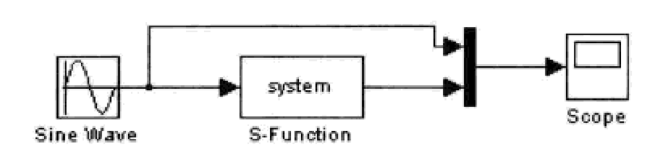


图 4-35　S-函数模型

2）在图 4-34 中双击“Level-1 M-file s-function Template”模块，系统弹出 S-函数模板源文件如下：

```
function [sys,x0,str,ts,simStateCompliance] = sfuntmpl(t,x,u,flag)
switch flag,
  case 0,
    [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;
  case 1,
    sys=mdlDerivatives(t,x,u);
  case 2,
    sys=mdlUpdate(t,x,u);
  case 3,
    sys=mdlOutputs(t,x,u);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end
function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 0;
sizes.NumInputs       = 0;
sizes.DirFeedthrough = 1;
```

```
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0   = [];
str = [];
ts   = [0 0];
simStateCompliance = 'UnknownSimState';
function sys=mdlDerivatives(t,x,u)
sys = [];
function sys=mdlUpdate(t,x,u)
sys = [];
function sys=mdlOutputs(t,x,u)
sys = [];
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
function sys=mdlTerminate(t,x,u)
sys = [];
```

上述 S-函数 M 文件的源代码中，已删除文件解释部分文字。

在本例题中，为了完成设定的函数功能 $y = 5u - 3$，可将上述 S-函数的模板进行如下修改（这里只列出需要修改的部分）。

首先，将 S-函数文件首先修改为：function[sys, x0, str, ts]= xiuMS (t, x, u,flag)，并将文件另存为 xiuMS.m 文件。

其次，找到[sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes，修改如下：

```
sizes.NumOutputs       = 1;
sizes.NumInputs        = 1;
```

然后，再找到 function sys=mdlOutputs(t,x,u)，修改如下：

```
sys = 5*u-3;
```

最后，将该文件保存到当前目录下的 xiuMS.m 文件中。

3）双击仿真模型中的 S-Function 模块，打开参数设置对话框，并修改 S-Function 的名称为“xiuMS”。

4）仿真参数采用默认设置，运行仿真，效果如图 4-36 所示。

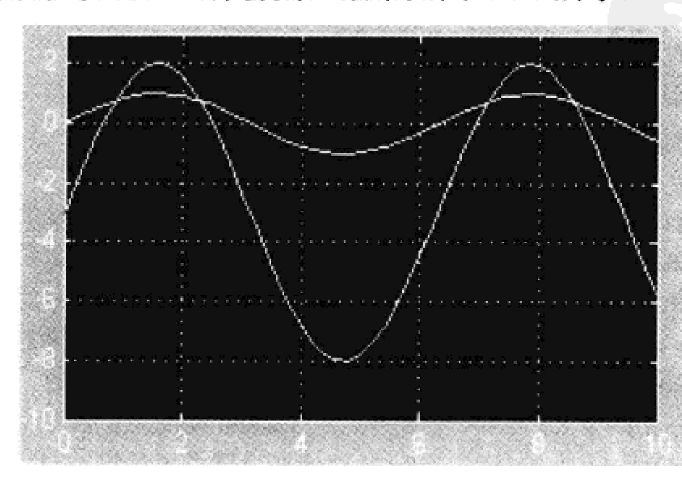


图 4-36　仿真结果



4.3.2 M 文件的 S-函数

1. M 文件的 S-函数工作过程

M 文件 S-函数源代码主要由 7 部分组成，如下所示（为了便于观察，已经删除源代码中的英文注释部分，适当添加了中文说明）。

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
switch flag,
%初始化函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%求导数
  case 1,
    sys=mdlDerivatives(t,x,u);
%状态更新
  case 2,
    sys=mdlUpdate(t,x,u);
%计算输出
  case 3,
    sys=mdlOutputs(t,x,u);
%计算下一个采样时刻
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
  case 9,
    sys=mdlTerminate(t,x,u);
%错误处理
  otherwise
    error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
% mdlInitializeSizes 模型初始化函数,返回:
% sys 是系统参数
% x0 是系统初始状态,若没有状态,取[ ]
% str 是系统阶字串,通常设为[ ]
% ts 是取样时间矩阵,对连续取样时间,ts 取[0 0]
% 若使用内部取样时间,ts 取[-1 0],-1 表示继承输入信号的采样周期
function [sys,x0,str,ts]=mdlInitializeSizes %模型初始化函数
sizes = simsizes;                              %取系统默认设置
sizes.NumContStates   = 0;                     %设置连续状态变量的个数
sizes.NumDiscStates   = 0;                     %设置离散状态变量的个数
sizes.NumOutputs      = 0;                     %设置系统输出变量的个数
sizes.NumInputs       = 0;                     %设置系统输入变量的个数
sizes.DirFeedthrough = 1;                      %设置系统是否直通
sizes.NumSampleTimes = 1;                      %采样周期的个数,必须大于或等于 1
sys = simsizes(sizes);                         %设置系统参数
x0  = [];                                      %系统状态初始化
```

```
str = [];                                  %系统阶字串总为空矩阵
ts  = [0 0];                               %初始化采样时间矩阵
% mdlDerivatives 模型计算导数——连续状态部分的计算，返回连续状态的导数
function sys=mdlDerivatives(t,x,u)
sys = [];          %根据状态方程（微分方程部分）修改此处
% mdlUpdate 计算离散状态部分
function sys=mdlUpdate(t,x,u)
sys = [];          %根据状态方程（差分方程部分）修改此处
% mdlOutputs 计算输出信号，返回模块的输出
function sys=mdlOutputs(t,x,u)
sys = [];          %根据输出方程修改此处
% mdlGetTimeOfNextVarHit 计算一步的仿真时刻，该函数仅当在 mdlInitializeSizes
%函数中的采样时间向量定义了一个可变离散采样时间 ts 为[-2 0]时才被使用
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;   %例如，下一步仿真时间是 1s 之后
sys = t + sampleTime;
% mdlTerminate 终止仿真设定，完成仿真终止时的任务
function sys=mdlTerminate(t,x,u)
sys = [];
% 程序结束
```



在初始化阶段，通过控制变量 flag=0 调用 S-函数，并请求提供输入/输出变量个数、初始状态和采样时间等信息。然后，仿真开始。通过修改控制变量 flag=4，请求 S-函数提供下一步的采样时间（对于固定采样时间的系统，此函数不被调用）。接下来修改控制变量 flag=3，计算块的输出。接着修改控制变量 flag=2，更新每一个采样时间的系统离散状态。对于连续系统，再修改控制变量 flag=1，求连续系统状态变量的导数。然后再通过控制变量 flag=3 计算新的块输出。这样就完成了一个仿真步长的计算工作。当仿真结束后，通过控制变量 flag=9，调用结束处理函数，进行结束前的处理工作。

下面以示例来说明 M 文件中 S-函数的编写及仿真过程。

【例 4-2】 已知某系统状态方程如下：

$$\dot{x}=\begin{pmatrix}-1 & 0 & 0\\ 0 & -2 & 0\\ 0 & 0 & -3\end{pmatrix}x+\begin{pmatrix}3\\ -22\\ 20\end{pmatrix}u,\quad y=[-1\quad -1\quad -1]x$$

试用 S-函数建立其仿真模型，并求其单位阶跃响应曲线。

首先打开 S-函数模板文件。打开 S-函数模板文件除了按照上述介绍的方法，也可以在 Command Windows 中输入以下代码：

```
>> open sfuntmp1
```

或者

```
>> edit sfuntmp1
```

执行命令后系统弹出 M 文件的 S-函数编辑窗口。删除注释信息后，实现上述系统的 S-

函数源程序代码如下所示：

```
function [sys,x0,str,ts]= li4_2(t, x, u,flag)
switch flag,
%初始化函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%求导数
  case 1,
    sys=mdlDerivatives(t,x,u);
%状态更新
  case 2,
    sys=mdlUpdate(t,x,u);
%计算输出
  case 3,
    sys=mdlOutputs(t,x,u);
%计算下一个采样时刻
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
  case 9,
    sys=mdlTerminate(t,x,u);
%错误处理
  otherwise
    error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes              %模型初始化函数
sizes = simsizes;                                         %取系统默认设置
sizes.NumContStates   = 3;                                %设置连续状态变量的个数
sizes.NumDiscStates   = 0;                                %设置离散状态变量的个数
sizes.NumOutputs      = 1;                                %设置系统输出变量的个数
sizes.NumInputs       = 1;                                %设置系统输入变量的个数
sizes.DirFeedthrough = 0;                                 %设置系统是否直通
sizes.NumSampleTimes = 1;                                 % 采样周期的个数，必须大于或等于 1
sys = simsizes(sizes);                                    %设置系统参数
x0   = [0;0;0];                                           %系统状态初始化
str = [];                                                 %系统阶字串总为空矩阵
ts   = [0 0];                                             %初始化采样时间矩阵
function sys=mdlDerivatives(t,x,u)
x(1)=-1*x(1)+3*u;
x(2)=-2*x(2)-22*u;
x(3)=-3*x(3)+20*u;
sys=x;
function sys=mdlUpdate(t,x,u)
sys = [];                  %根据状态方程（差分方程部分）修改此处
function sys=mdlOutputs(t,x,u)
sys =-x(1)-x(2)-x(3);      %根据输出方程修改此处
```



```
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;      %  例如，下一步仿真时间是 1s 之后
sys = t + sampleTime;
% mdlTerminate 终止仿真设定，完成仿真终止时的任务
function sys=mdlTerminate(t,x,u)
sys = [];

% 程序结束
```

将上述 S-函数在当前目录下保存为 M 文件 xiuMli4_2，再建立 Simulink 模型，如图 4-37 所示，并设置 S-Function 模块的参数为 S-function name，仿真参数取默认值，运行仿真效果如图 4-38 所示。



图 4-37　S-函数仿真模型

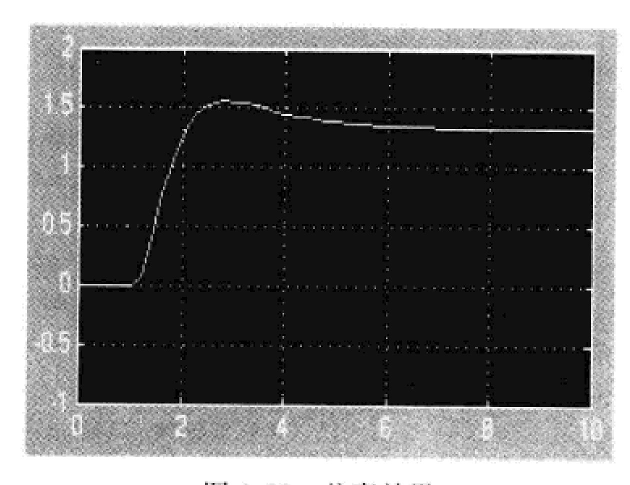


图 4-38　仿真效果

2. 在 S-函数中添加用户参数

当 S-函数要添加用户参数时，要注意两点：① 在 S-函数源代码中，用到该参数的各个子函数，在函数声明部分均应添加该参数；② 在 Simulink 模型中设置“S-Function”模块参数时，参数的名称和顺序必须与 S-函数源代码中的参数名称和顺序完全一致。下面以一个三角波发生器为例，来学习如何在 S-函数中添加用户参数。

【例 4-3】 试建立 Simulink 下三角波函数发生器，且三角波频率幅值可调。

首先编写成三角波 S-函数的 M 文件，打开模板程序，其修改如下：

```
function [sys,x0,str,ts]= xiuMli4_3(t,x,u,flag,a,freq)
%输入参数 a 为三角波幅值，freq 为三角波频率
switch flag,
%初始化函数
  case 0,
     [sys,x0,str,ts]=mdlInitializeSizes;
%求导数
  case 1,
```

```
        sys=mdlDerivatives(t,x,u);
%状态更新
    case 2,
        sys=mdlUpdate(t,x,u);
%计算输出
    case 3,
%因 mdlOutputs 函数部分将用到附加参数 a 和 freq，所以在输入参数列表应添加该参数
        sys=mdlOutputs(t,x,u,a,freq);
%计算下一个采样时刻
    case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
    case 9,
        sys=mdlTerminate(t,x,u);
%错误处理
    otherwise
        error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes          %模型初始化函数
sizes = simsizes;                                   %取系统默认设置
sizes.NumContStates   = 0;                          %设置连续状态变量的个数
sizes.NumDiscStates   = 0;                          %设置离散状态变量的个数
sizes.NumOutputs      = 1;                          %设置系统输出变量的个数
sizes.NumInputs       = 1;                          %设置系统输入变量的个数
sizes.DirFeedthrough = 1;                           %设置系统是否直通
sizes.NumSampleTimes = 1;                           %采样周期的个数，必须大于或等于 1
sys = simsizes(sizes);                              %设置系统参数
x0   = [];                                          %系统状态初始化
str = [];                                           %系统阶字串总为空矩阵
ts   = [0 0];                                       %初始化采样时间矩阵
function sys=mdlDerivatives(t,x,u)
sys=[];
function sys=mdlUpdate(t,x,u)
sys = []; %根据状态方程（差分方程部分）修改此处
function sys=mdlOutputs(t,x,u,a,freq)
%直接在输出函数部分编写生成三角波的源代码
T=1/freq;          %求周期
m=rem(u,T);        %u 为外部输入时间信息，rem 为求余函数
k=floor(u/T);      %floor 为向零取整
R=4*a*freq;
c=T/2;
if((m>=0)&(m<c))
    sys=R*(u-(k+0.25)*T);
elseif((m>=c)&(m<=T))
    sys=-[R*(u-(k+0.75)*T)];
else
```

```
        sys=a;
    end
    % 程序结束
```



将上述 M 文件的 S-函数在当前目录下保存为 xiuMli4_3.m 文件，建立 Simulink 仿真模型，如图 4-39 所示。双击“S-Function”模块，在弹出的参数设置对话框中，设置“S-function name”文本框为“xiuMli4_3”，设置“S-function parameters”文本框为“1，1500”，如图 4-40 所示。将 Simulink 求解器参数“Max step size”设置为 1e-5，“Stop time”设置为 0.2s，其他取系统默认值。单击 Simulink 模型工具栏中的“Run”快捷按钮，运行仿真，结果如图 4-41 所示。

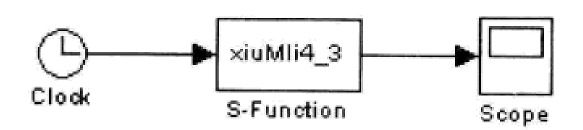


图 4-39　三角波函数发生器模型

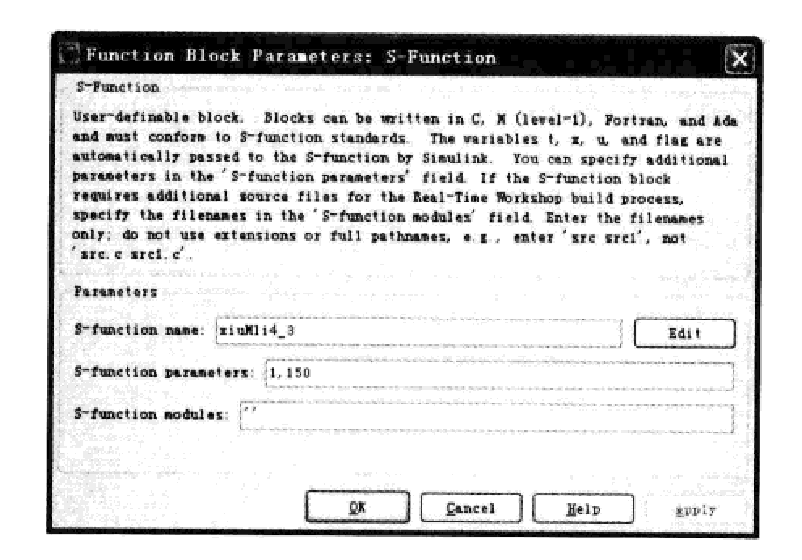


图 4-40　S-函数仿真参数设置

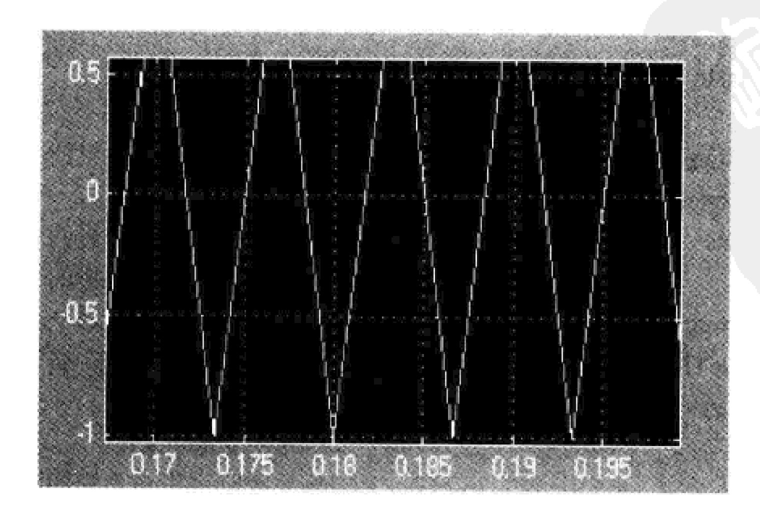


图 4-41　三角波函数发生器的 S-函数仿真波形

3. 离散系统的 S-函数

离散系统的 S-函数仿真，需要使用到 flag=2，mdlUpdata(t, x, u)函数。要详细了解请看下示例。

【例 4-4】 已知离散 PID 表达式为：

$$U(k) = Kp \times e(k) + Ki \times T \times \sum_{m=0}^{k} e(k) + Kd \times [e(k-1) - e(k-2)]/T$$

试使用 S-函数实现离散 PID 控制器，并建立实现其 Simulink 模型。

首先打开 S-函数模板程序，建立离散 PID 的 S-函数源文件，其源代码如下：

```
function [sys,x0,str,ts]= xiuMli4_4(t,x,u,flag,kp,ki,kd)
switch flag,
%初始化函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%求导数
  case 1,
    sys=mdlDerivatives(t,x,u);
%状态更新
  case 2,
    sys=mdlUpdate(t,x,u,kp,ki,kd);
%计算输出
  case 3,
    sys=mdlOutputs(t,x,u,kp,ki,kd);
%计算下一个采样时刻
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
  case 9,
    sys=mdlTerminate(t,x,u);
%错误处理
  otherwise
    DAStudio.error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes          %模型初始化函数
sizes = simsizes;                                    %取系统默认设置
sizes.NumContStates    = 0;                          %设置离散状态变量的个数
sizes.NumDiscStates    =4;                           %离散系统输出个数
sizes.NumOutputs       = 1;                          %设置系统输出变量的个数
sizes.NumInputs        = 1;                          %设置系统输入变量的个数
sizes.DirFeedthrough = 0;                            %设置系统是否直通
sizes.NumSampleTimes = 1;                            %采样周期的个数，必须大于或等于 1
sys = simsizes(sizes);                               %设置系统参数
x0   = [0;0;0;0];                                    %系统状态初始化
str = [];                                            %系统阶字串总为空矩阵
ts   = [-2 0];
```



```
%ts(1)=-2 表示采样时间由 flag=4 和 mdlGetTimeOfNextVarHit(t,x,u)决定下一个采样时刻
function sys=mdlDerivatives(t,x,u)
sys=[];
function sys=mdlUpdate(t,x,u,kp,ki,kd)
x(3)=x(2);
x(2)=x(1);
x(1)=u;
x(4)=u+x(4);
sys=x;
function sys=mdlOutputs(t,x,u,kp,ki,kd)
sys=kp*x(1)+ki*0.01*x(4)+kd*(x(2)-x(3))/0.01;
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime =0.01;      %设置采样周期
sys = t + sampleTime;
% mdlTerminate 终止仿真设定，完成仿真终止时的任务
function sys=mdlTerminate(t,x,u)
sys = [];
% 程序结束
```

将上述 M 文件在当前目录下保存为“xiuMli4_4.m”。建立 Simulink 仿真模型，如图 4-42 所示。双击“S-Function”模块，系统弹出其参数设置对话框，如图 4-43 所示。在图 4-43 中，设置“S-function name”文本框为“xiuMli4_4”，设置“S-function parameters”文本框为“6.0, 5.0, 0.5”。Simulink 仿真参数取默认值，运行仿真效果如图 4-44 所示。

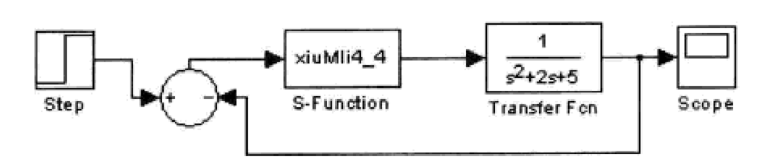


图 4-42　离散 PID 系统仿真

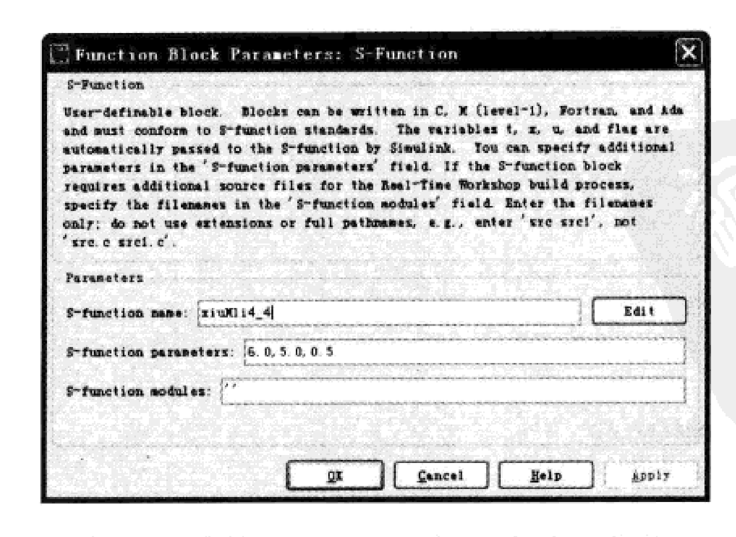


图 4-43　离散 PID-Function 模块参数设置对话框

4. 连续系统的 S-函数

连续系统的数学模型通常用微分方程、传递函数或状态方程来描述，用 M 文件的 S-函

数描述时，需要将其转换为状态方程的形式。下面以实例说明用 S-函数实现连续系统建模的基本方法。

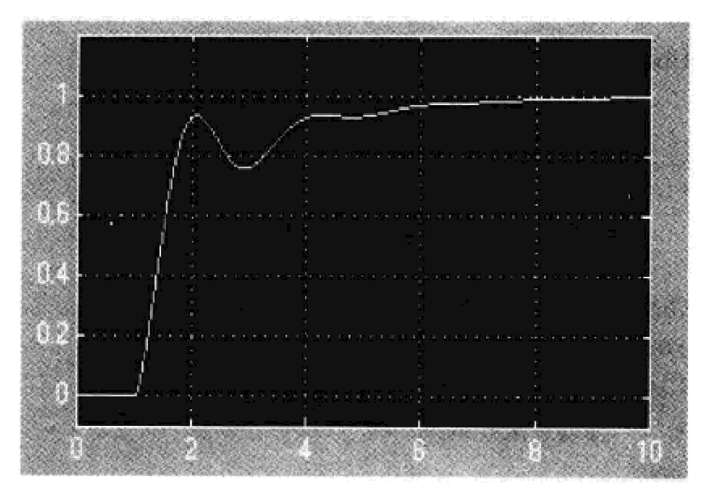


图 4-44　离散 PID S-函数仿真波形

【例 4-5】 已知系统微分方程为：$\dddot{y}+2\ddot{y}+10\dot{y}+5y=10u$ 。式中，u、y 分别为系统的输入量和输出量，且输出量 y 的各阶导数为零。使用 S-函数加以实现并求其单位阶跃响应。

首先把微分方程转换为状态方程，其结果如下。

系统状态方程为：$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix}=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 2\\ -5 & -10 & -2\end{pmatrix}x+\begin{pmatrix}0\\ 0\\ 10\end{pmatrix}u$

输出方程为：$y=[1\quad 0\quad 0]x$

然后打开 S-函数模板程序，并进行相应修改，其源程序代码如下：

```
function [sys,x0,str,ts]= xiuMli4_5(t,x,u,flag)
switch flag,
%初始化函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%求导数
  case 1,
    sys=mdlDerivatives(t,x,u);
%状态更新
  case 2,
    sys=mdlUpdate(t,x,u);
%计算输出
  case 3,
    sys=mdlOutputs(t,x,u);
%计算下一个采样时刻
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
  case 9,
    sys=mdlTerminate(t,x,u);
```





```
%错误处理
  otherwise
    error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes          %模型初始化函数
sizes = simsizes;                                    %取系统默认设置
sizes.NumContStates    = 3;                          %设置连续状态变量的个数
sizes.NumDiscStates    = 0;                          %设置离散状态变量的个数
sizes.NumOutputs       = 1;                          %设置系统输出变量的个数
sizes.NumInputs        = 1;                          %设置系统输入变量的个数
sizes.DirFeedthrough = 0;                            %设置系统是否直通
sizes.NumSampleTimes = 1;                            %采样周期的个数，必须大于或等于 1
sys = simsizes(sizes);                               %设置系统参数
x0    = [0;0;0];                                     %系统状态初始化
str = [];                                            %系统阶字串总为空矩阵
ts    = [0 0];                                       %初始化采样时间矩阵
function sys=mdlDerivatives(t,x,u)
x(3)=-5*x(1)-10*x(2)-6*x(3)+10*u;
x(1)=x(2);
x(2)=x(3);
sys=x;
function sys=mdlUpdate(t,x,u)
sys = []; %根据状态方程（差分方程部分）修改此处
function sys=mdlOutputs(t,x,u)
sys=x(1);
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;      %例如，下一步仿真时间是 1s 之后
sys = t + sampleTime;
% mdlTerminate 终止仿真设定，完成仿真终止时的任务
function sys=mdlTerminate(t,x,u)
sys = [];
% 程序结束
```

将上述 S-函数在当前目录下保存为 M 文件“xiuMli4_5.m”。建立 Simulink 模型，如图 4-45 所示。双击“S-Function”模块，在弹出的参数设置对话框，设置“S-function name”为“continuousfun”，其他参数取系统默认值。运行仿真，其运行仿真如图 4-46 所示。

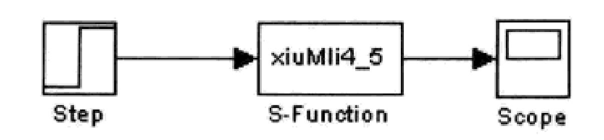


图 4-45 连续系统仿真模型

4.3.3 S-函数示例

【例 4-6】 试用 S-函数实现一个连续积分器。

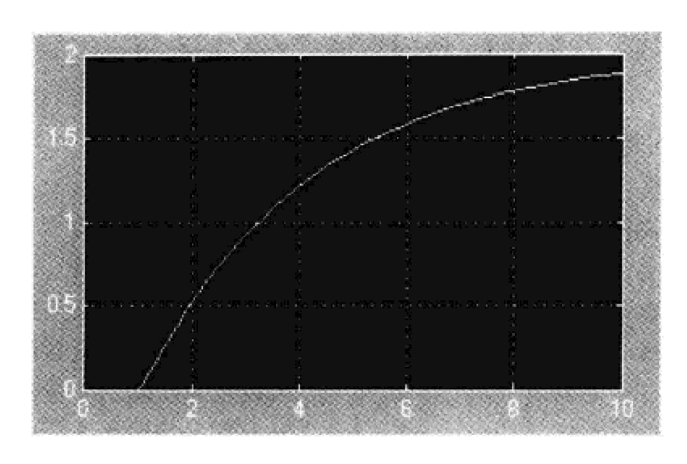


图 4-46 连续系统仿真波形

连续积分器数学模型为 $y=\int u\mathrm{d}t$ ，即 $\dot{y}=u$ 。设 $y=x$ ，则积分器状态方程为 $\dot{x}=u$ ，输出方程为 $y=x$ 。打开 S-函数模板，其源代码修改如下：

```
function [sys,x0,str,ts]= xiuMli4_6(t,x,u,flag,xiu_state)
switch flag,
%初始化函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(xiu_state);
%求导数
  case 1,
    sys=mdlDerivatives(t,x,u);
%状态更新
  case 2,
    sys=mdlUpdate(t,x,u);
%计算输出
  case 3,
    sys=mdlOutputs(t,x,u);
%计算下一个采样时刻
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
  case 9,
    sys=mdlTerminate(t,x,u);
%错误处理
  otherwise
    error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes(xiu_state) %模型初始化函数
sizes = simsizes;                               %取系统默认设置
sizes.NumContStates  = 1;                       %设置连续状态变量的个数
sizes.NumDiscStates  = 0;                       %设置离散状态变量的个数
sizes.NumOutputs     = 1;                       %设置系统输出变量的个数
sizes.NumInputs      = 1;                       %设置系统输入变量的个数
sizes.DirFeedthrough = 0;                       %设置系统是否直通
sizes.NumSampleTimes = 1;                       %采样周期的个数，必须大于或等于 1
```



```
sys = simsizes(sizes);                          %设置系统参数
x0   = [xiu_state];                             %系统状态初始化
str = [];                                       %系统阶字串总为空矩阵
ts   = [0 0];                                   %初始化采样时间矩阵
function sys=mdlDerivatives(t,x,u)
x=u
sys=x;
function sys=mdlUpdate(t,x,u)
sys = []; %根据状态方程（差分方程部分）修改此处
function sys=mdlOutputs(t,x,u)
sys=x;
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;      %例如，下一步仿真时间是 1s 之后
sys = t + sampleTime;
% mdlTerminate 终止仿真设定，完成仿真终止时的任务
function sys=mdlTerminate(t,x,u)
sys = [];
% 程序结束
```

将上述 S-函数保存为 M 文件 xiuMli4_6.m，然后建立 Simulink 仿真模型，如图 4-47 所示，双击“S-Function”模块，在弹出的参数设置对话框中设置“S-function name”为“xiuMli4_6”，设置“S-function parameters”文本框为“0”。Simulink 求解器的参数取默认值。运行仿真，结果如图 4-48 所示。从图中可以看出，采用 S-函数实现的连续积分器对正弦输入信号进行积分，结果为余弦函数，符合数学定义式。

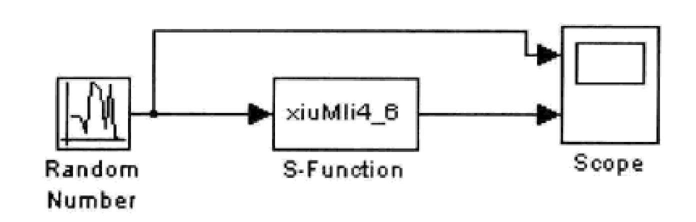


图 4-47　连续积分器 S-函数仿真模型

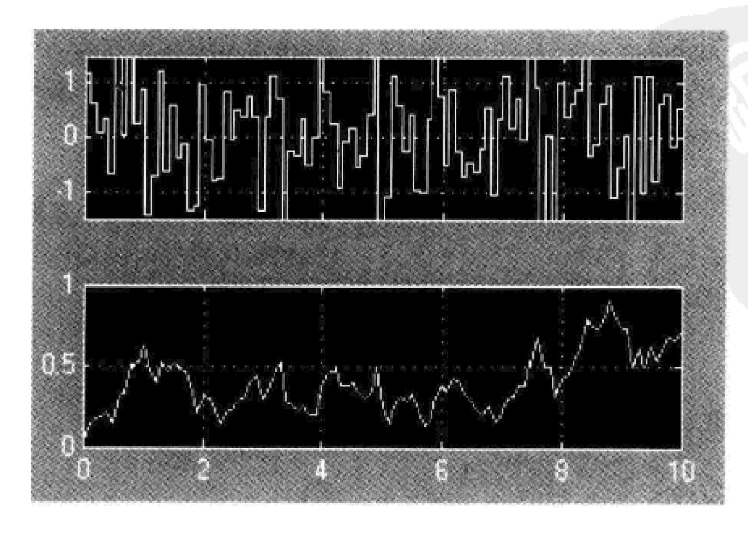


图 4-48　S-函数的连续积分对随机信号积分

【例 4-7】 试用 S-函数实现一个离散系统的单位延迟环节。

打开 S-函数的模板文件，其源程序修改如下：

```
function [sys,x0,str,ts]= xiuMli4_7(t,x,u,flag)
switch flag,
%初始化函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
%求导数
  case 1,
    sys=mdlDerivatives(t,x,u);
%状态更新
  case 2,
    sys=mdlUpdate(t,x,u);
%计算输出
  case 3,
    sys=mdlOutputs(t,x,u);
%计算下一个采样时刻
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
  case 9,
    sys=mdlTerminate(t,x,u);
%错误处理
  otherwise
    error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes          %模型初始化函数
sizes = simsizes;                                     %取系统默认设置
sizes.NumContStates   = 0;                            %设置连续状态变量的个数
sizes.NumDiscStates   = 1;                            %设置离散状态变量的个数
sizes.NumOutputs      = 1;                            %设置系统输出变量的个数
sizes.NumInputs       = 1;                            %设置系统输入变量的个数
sizes.DirFeedthrough = 0;                             %设置系统是否直通
sizes.NumSampleTimes = 1;                             %采样周期的个数，必须大于或等于 1
sys = simsizes(sizes);                                %设置系统参数
x0   = [0];                                           %系统状态初始化
str = [];                                             %系统阶字串总为空矩阵
ts   = [-2 0];                                        %初始化采样时间矩阵
function sys=mdlDerivatives(t,x,u)
x=[]
sys=x;
function sys=mdlUpdate(t,x,u)
x=u;
sys = x;
function sys=mdlOutputs(t,x,u)
```

```
sys=x;
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 0.1;
sys = t + sampleTime;
% mdlTerminate 终止仿真设定，完成仿真终止时的任务
function sys=mdlTerminate(t,x,u)
sys = [];
% 程序结束
```

将上述 S-函数在当前目录下保存为 M 文件 xiuMli4_7.m，然后建立 Simulink 模型，如图 4-49 所示，双击“S-Function”模块，在弹出的参数设置对话框中设置“S-function name”文本框为“xiuMli4_7”。离散系统 S-函数的采样时间由 flag=4 的回调函数来设置，取 T=0.1s，Simulink 求解器的参数取默认值。运行仿真，效果如图 4-50 所示。

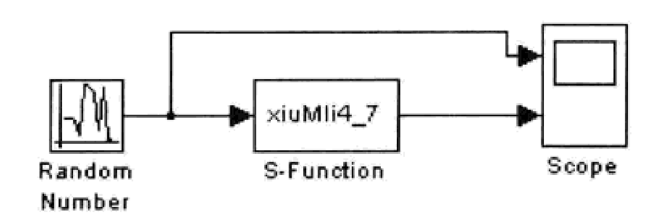


图 4-49 单位延迟离散 S-函数仿真模型

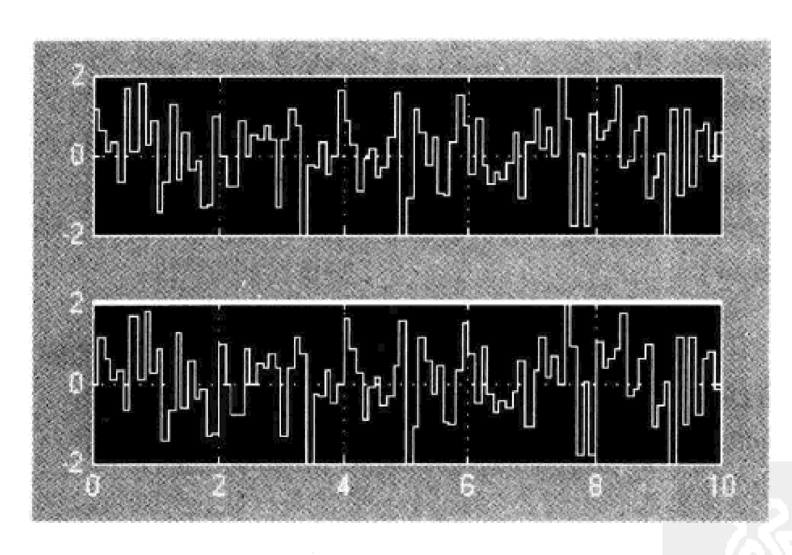


图 4-50 延迟一个采样周期输出的离散 S-函数仿真波形

【例 4-8】 试用 S-函数实现一个连续积分系统，外加一个离散的单位延迟。

这是一个连续、离散混合系仿真实例，打开 S-函数模板，其源代码修改如下：

```
function [sys,x0,str,ts]= xiuMli4_8(t,x,u,flag)
Dper=0.4;    %设置离散采样周期
Doff=0;      %设置采样偏移时间
switch flag,
%初始化函数
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(Dper,Doff);
```



```
%求导数
  case 1,
    sys=mdlDerivatives(t,x,u);
%状态更新
  case 2,
    sys=mdlUpdate(t,x,u,Dper,Doff);
%计算输出
  case 3,
    sys=mdlOutputs(t,x,u,Dper,Doff);
%计算下一个采样时刻
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
%终止仿真程序
  case 9,
    sys=mdlTerminate(t,x,u);
%错误处理
  otherwise
    error(['Simulink:blocks:unhandledFlag', num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes(Dper,Doff) %模型初始化函数
sizes = simsizes;                              %取系统默认设置
sizes.NumContStates  = 1;                      %设置连续状态变量的个数
sizes.NumDiscStates  = 1;                      %设置离散状态变量的个数
sizes.NumOutputs     = 1;                      %设置系统输出变量的个数
sizes.NumInputs      = 1;                      %设置系统输入变量的个数
sizes.DirFeedthrough = 0;                      %设置系统是否直通
sizes.NumSampleTimes = 2;                      %本系统为混合系统，有两种采样时间
sys = simsizes(sizes);                         %设置系统参数
x0  = [0;0];                                   %系统状态初始化
str = [];                                      %系统阶字串总为空矩阵
ts  = [0,0;Dper,Doff];                         %初始化采样时间矩阵
function sys=mdlDerivatives(t,x,u)
x(1)=u;
sys=x(1);
function sys=mdlUpdate(t,x,u,Dper,Doff)
if abs((t-Doff)/Dper-round((t-Doff)/Dper))<1e-8
    x(2)=x(1);
    sys=x(2);
else
    sys=[];
end
function sys=mdlOutputs(t,x,u,Dper,Doff)
if abs((t-Doff)/Dper-round((t-Doff)/Dper))<1e-8
    sys=x(2);
else
```



```
    sys=[];
end
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 0.5;
sys = t + sampleTime;
% mdlTerminate 终止仿真设定，完成仿真终止时的任务
function sys=mdlTerminate(t,x,u)
sys = [];
% 程序结束
```

将上述 S-函数保存为 M 文件 xiuMli4_8.m，然后建立如图 4-51 所示的积分加单位延迟的混合系统 S-函数仿真模型。双击“S-Function”模块，在弹出的参数设置对话框中设置“S-function name”文本框为“xiuMli4_8”，设置 Simulink 仿真求解器的“Max step size”（最大仿真步长）为 0.01，其他取默认值，运行仿真得如图 4-52 所示的混合仿真结果。

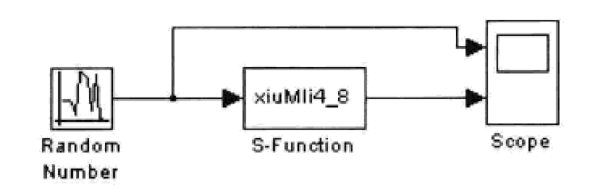


图 4-51 积分加单位延迟的混合系统 S-函数仿真模型

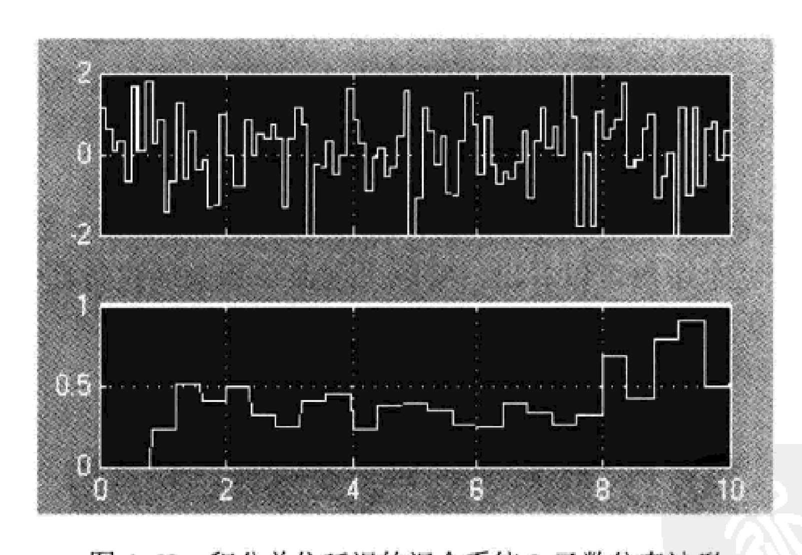


图 4-52 积分单位延迟的混合系统 S-函数仿真波形

4.4 Simulink 的命令仿真

4.4.1 使用命令创建 Simulink 仿真模型

从广义上说，凡是涉及命令行调用实现 Simulink 仿真的 MATLAB 命令，都可以称为是命令行 Simulink 仿真，MATLAB 提供了丰富的命令与 Simulink 工具箱进行接口，方便进行命令行创建与仿真 Simulink。下面让读者来阅读一段程序段：

```
>> %创建模型,并设置参数,进行仿真
>> new_system('command_sim_demo2')
>> open_system('command_sim_demo2');
>> add_block('built-in/step','command_sim_demo2/step');
>> add_block('Simulink/Continuous/Transfer Fcn','command_sim_demo2/2-order system');
>> add_block('built-in/scope','command_sim_demo2/scope');
>> add_line('command_sim_demo2','step/1','2-order system/1');
>> add_line('command_sim_demo2','2-order system/1','scope/1','autorouting','on');
>> save_system('command_sim_demo2','command_create_demo2');
>>%设置模块参数和仿真参数
>> set_param('command_create_demo2/step','Time','0','Before','0','After','1','SampleTime','0');
>> set_param('command_create_demo2/2-order system','Numerator','[100]','Denominator','[1 141.4 100]');
>> set_param('command_create_demo2/scope','NumInputPorts','1');
>> set_param('command_create_demo2','Solver','ode45','StopTime','9');
>> %保存后进行仿真
>> save_system
>> sim('command_create_demo2');
```

相信读者通过一些函数的命名和程序注释已经明白这段程序的功能。创建文件名为 command_sim_demo2 的仿真模型，并且将其保存为 command_create_demo2。然后设置模块参数和仿真参数，该仿真模型包含一个阶跃信号源、传输函数模块和示波器模块，阶跃信号设置阶跃时间为 0，阶跃值为 1，传递函数模块分子多项式为 100，而分母多项式为[1 141.4 100]，设置示波器的输入端口数目为 1 个，最后设置模型的仿真参数，采用变步长 ode45 算法，仿真结束时间为 10s。经过模块参数和仿真参数设置后，保存仿真模型，并且进行仿真。

如图 4-53 所示为命令行建立的仿真模型和仿真结果，并保存了名为 command_create_demo2.mdl 模型。从这个示例可以看出，实际上用命令行来创建仿真模型是比较繁琐的事情，对于如图 4-53 所示的简单仿真模型，需要编写大量的程序代码，而且需要用户熟悉各个模块所在 Simulink 中的位置，效果如图 4-54 所示。

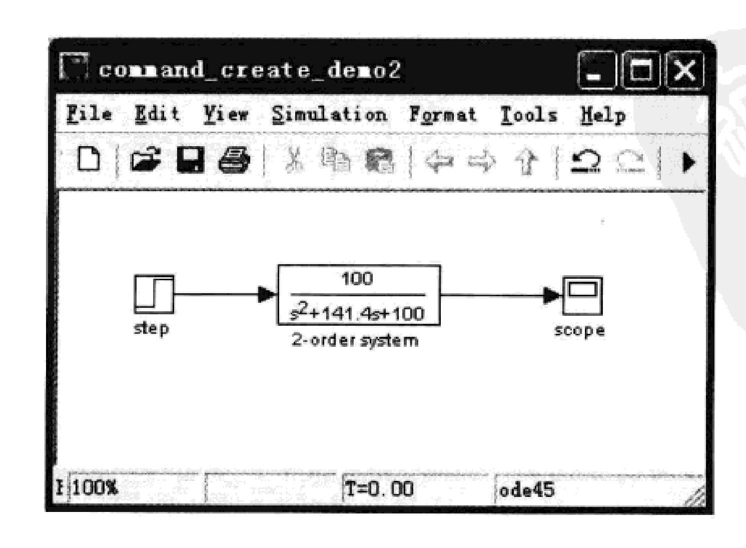


图 4-53　命令行建立仿真模型

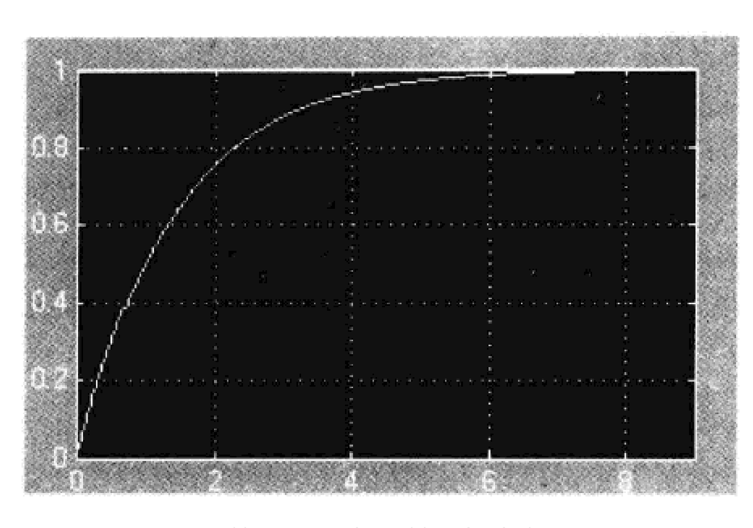


图 4-54 命令行仿真结果

虽然可以使用 Build-in 命令调用 Simulink 任何模块库中的文件，但是用户还必须熟悉各模块的名称，这对于用户而言，要记住每个模块的名称，显然很不现实，而且确实也没有这样的必要性。

通过图形化的 Simulink 仿真平台进行模块的拖放和连线、属性设置是比较方便实用的。因此对用命令创建仿真模型不用详细介绍，在实际应用中，用得比较少。上面的程序段只是向读者演示一下如何使用命令行创建和仿真模型，以便让读者心中有数。对于 set_param 和 get_param 两条命令在后续命令行仿真技术介绍中用得比较广泛，因此对于这两条 Simulink 仿真命令作详细介绍。

（1）set_param 命令

set_param 命令用来设置仿真模型的参数或者仿真模型中具体模块的参数。其调用格式如下：

```
set_param('obj','parameter1','value1','parameter2','value2',…)%设置仿真模型或者模块的参数
%将仿真模型参数设置为默认值
set_param('0','modelparm1 ','value1',' modelparm2','value2',…)
```

第一条语句允许用户设置仿真模型的属性，包括仿真算法设置等，同时也允许用户设置具体仿真模块的参数。通过使用 get_param 命令来获取仿真模块属性名称和仿真算法属性名称。

第二条语句允许用户将仿真模型参数设置为默认值，即改变 MATLAB 默认的 Simulink 仿真参数，在新建仿真模型后，那么模型的默认参数值就是用户自己设定的 Simulink 仿真参数。为了在获取仿真参数时，可以使用 simset 命令得到仿真模型的仿真结构参数，包括属性名称和允许的属性值。

```
>> %设置传递函数模块,分子多项式为[100],分母多项式为[1 141.4 100]
set_param('command_create_demo2/2-order system','Numerator','[100]','Denominator','[1,141.4,100]');
%设置仿真模型的仿真参数
set_param('command_create_demo2','Solver','ode45','StopTime','9');
%设置仿真模型默认参数:采用 ode23 算法,初始步长为 1e-3
set_param(0,'solver','ode23','Initialstep','1e-3');
```





```
%设置仿真模块的位置
set_param('command_create_demo2/step','position',[50 100 110 120]);
%设置仿真模块的回调函数
set_param('command_create_demo2/step','OpenFcn','Open_Fcn','closeFcn','close_fcn');
```

注意：模块或仿真模型参数属性值通常是字符串形式，从上面的示例可以看到 position 属性的属性值是数组，另一个非字符串属性值的属性 UserData 同样是用户自定义数据类型。

（2）get_param 命令

get_param 命令可以获取指定模块的属性值、当前仿真模型的默认参数，以及仿真模块属性的结构体，即模块的属性名称。其调用格式如下：

```
get_param('obj','parameter')%获取仿真模型参数和特定仿真模块参数属性值
get_param({objects},'parameter')%获取多个模块参数属性值,模块路径以{}元胞形式定义
get_param(handle,'parameter')%返回句柄 handle 对象的参数属性值
get_param(0,'parameter')%获取当前仿真模型或者模块的默认属性值
get_param('obj','ObjectParameters')%获取对象参数的结构体
get_param('obj','DialogParameters')%获取仿真模块的参数名称结构体
以下通过具体的命令形式和注释演示这些命令的调用形式：
%获取仿真模型求解算法属性值
>> get_param('command_create_demo2','Solver')
ans =
ode45
%获取 step 模块的仿真时间属性值
>> get_param('command_create_demo2/step','SampleTime')
>> param=get_param({'command_create_demo2/step','command_create_demo2/scope'},...
'DialogParameters')
param =
    [1x1 struct]
    [1x1 struct]
%可以分别使用 param{1}和 param{2}查看 step 模块和 scope 模块的对话框属性名称
%查看仿真模型中所包含的模块类型
>> blocks=find_system(gcs,'Type','block')
blocks =
    'command_create_demo2/2-order system'
    'command_create_demo2/scope'
    'command_create_demo2/step'
>> listblks=get_param(blocks,'BlockType')
listblks =
    'TransferFcn'
    'Scope'
    'Step'
%获取 step 模块对话框参数
>> get_param('command_create_demo2/step','DialogParameters')
ans =
```

```
            Time: [1x1 struct]
          Before: [1x1 struct]
           After: [1x1 struct]
      SampleTime: [1x1 struct]
  VectorParams1D: [1x1 struct]
       ZeroCross: [1x1 struct]
```

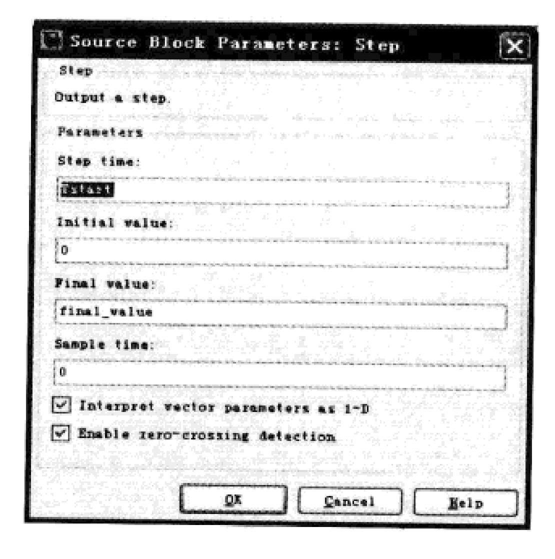


4.4.2 Simulink 命令行仿真技术

前面已简单介绍了使用命令行创建 Simulink 仿真模型的过程，可以看出对于一个非常简单的 Simulink 仿真模型，如果模块选择、模块间的连接、模块参数设置和仿真模型参数设置等一系列的操作完全使用命令行来完成时，编写相应的程序代码将非常繁琐，因此不建议使用命令行来创建 Simulink 仿真模型，直接使用 Simulink 仿真平台的图形可视化界面，在 Simulink Library Browser 窗口中进行模块的拖放、连线和属性设置、仿真参数设置，这样的过程是比较简单实用的。

Simulink 命令行仿真的优势在于能够重复地进行仿真模型的仿真，动态地改变仿真模型和模块的参数，记录多次仿真模型的结果，并进行数据分析。

在介绍仿真技术前，先通过一个示例向读者演示一些仿真技巧，仿真模型采用如图 4-53 所示的仿真模型。仿真模型中的 step 模块和 2-order system 模块的参数设置如图 4-55 所示，由于采用命令行进行仿真，相关的参数以变量形式表示，可以动态地改变仿真过程模块参数。仿真过程需要对每次仿真结果保存，那么需要设置示波器的属性如图 4-56 所示，保存变量名为 sim_out，数据格式为 Structure with time，这样就可以记录保存每次仿真过程的仿真时间和系统输出。



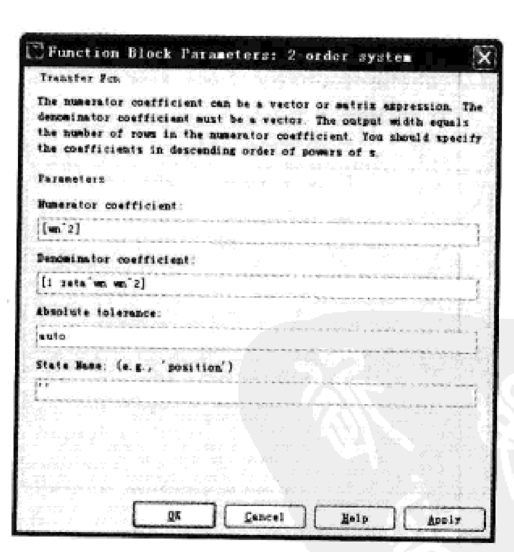


图 4-55 step 模块和 2-order system 模块的参数设置

建立好如图 4-53 所示的仿真模型后，需要使用命令行仿真技术实现在 zeta=0.707、wn=[4, 6, 8, 10]时对应的系统输出，并将它们同时画在一个图上，比较系统输出的差异。

```
>> Tstart=0;
final_value=1;
zeta=0.707;
```

图 4-56　示波器模块属性设置

```
wn_ref=[4 6 8 10];
set_param('command_create_demo2','Solver','ode45','stoptime','4');
%设置仿真模型的算法和终止时间
set_param('command_create_demo2/step','Time','0','After','1');
for i=1:length(wn_ref);
    sim('command_create_demo2');
    time{1,i}=sim_out.time;  %保存仿真时间
    out_value{1,i}=sim_out.signals.values;  %保存输出结果
end
%结果对比显示
colorstyle={'r-','k-','b-','m-'};
for i=1:length(time)
    plot(time{1,i},out_value{1,i},colorstyle{i});
    hold on;
end
```

仿真结果如图 4-57 所示。在 Simulink 仿真中，仿真模型可以直接从 MATLAB 的 Workspace 中调入参数进行仿真。从这下示例中读者可体会到：当仿真过程模型参数动态变化时，利用命令行仿真技术可以非常简单、快捷地实现模型参数动态变化、仿真结果的对比分析等。

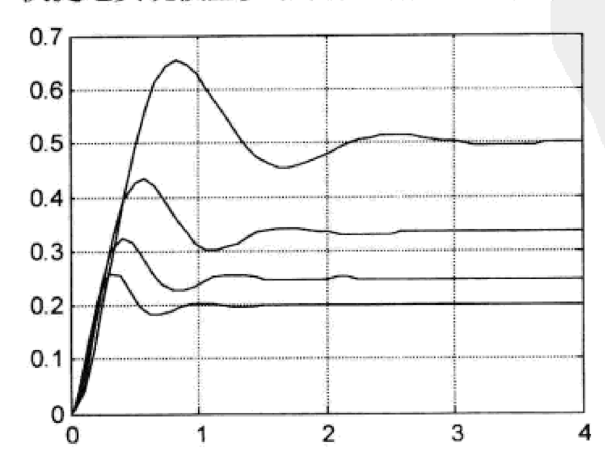


图 4-57　命令行仿真显示波形

在使用命令行进行动态系统仿真中，使用较为频繁的几个命令为：sim、simset、simget 和 simplot。下面将分别对它们展开介绍。

（1）sim 命令

sim 命令可以对指定的系统模型按照给定的仿真参数和模型参数进行动态仿真。其调用格式如下：



```
[t, x, y]=sim(model, timespan, options, ut);
[t, x, y1, y2, ..., yn]=sim(model, timespan, options ut);
```

函数输入/输出参数说明：以上两条命令是 sim 指令完整的调用格式，在通常情况下，除了 mode 参数外，其他参数可以是空集，使用系统模型的默认属性设置，包括系统配置参数和模型参数。model 参数为指定的仿真模型名称。

timespan 为仿真时间设置选项，可以有以下几种仿真时间设置方法。

- imespan=Tfinal：设置为终止时间 tFinal，起始时间默认为 0。
- timespan=[Tstart, Tfinal]：设置仿真模型的起始时间 Tstart 和终止时间 Tfinal。
- timespan=[Tstart, OutuputTimes, Tfinal]：设置仿真模型的起始时间 Tstart 和终止时间 Tfinal，同时设置仿真过程中输出的时间向量 OutputTimes。

前两种仿真模型时间设置输出的时间向量由仿真算法和仿真模型来确定，而第三种时间设置规定了仿真模型在哪些点上进行仿真输出。

options 选项包括了除仿真时间外的所有模型参数的设置，包括仿真算法选择、步长设置、相对误差和绝对误差设置、输入/输出数据名称和格式设置等选项，由 simset 进行设置。

ut 为外部变量的输入，可以是多个外部变量，格式是 $n \times 2$ 矩阵，第一列对应输入变量的时间，第二列对输入变量的值。

输出变量包括系统仿真时间向量 t、系统仿真状态变量矩阵 x 和系统仿真的输出矩阵。

```
>> %采用所有默认仿真参数和时间仿真模型
[t,x,y]=sim('command_create_demo2');
%设置仿真终止时间为 5s,仿真输出时间向量由默认求解算法决定
[t,x,y]=sim('command_create_demo2',5);
%设置仿真时间起始时间为 0,终止时间为 5s,时间间隔为 0.01s
[t,x,y]=sim('command_create_demo2',[0:0.01:5]);
%设置仿真参数最大步长 1e-4,数据保存名称 out_value,数据保存格式为 struct
options=simget('command_create_demo2');
simset(options,'MaxStep','1e-3','SaveFormat','Structure','OutputVariables','ty');
[t,x,y]=sim('command_create_demo2',[],options)
```

（2）simget 命令

同 set_param 命令和 get_param 命令一样，simget 命令和 simset 命令分别表示获取模型除仿真时间外的所有其他参数和设置模型的仿真参数。其调用格式如下：

```
struct = simget(model)
value = simget(model, 'param')
value = simget(OptionStructure, param)
```

第一条语句获取仿真模型的仿真参数结构体，不包含仿真时间参数；第二条语句获取仿真模型指定属性的属性值；第三条语句获取仿真模型结构体选项中指定属性的属性值。仿真模型的仿真参数结构体可以通过下面的指令获取。

```
>> options=simget('command_create_demo2')
options =
                          AbsTol: 'auto'
                           Debug: 'off'
                      Decimation: 1
                    DstWorkspace: 'current'
                  FinalStateName: ''
                       FixedStep: 'auto'
                    InitialState: []
                     InitialStep: 1.0000e-003
                        MaxOrder: 5
      ConsecutiveZCsStepRelTol: 2.8422e-013
             MaxConsecutiveZCs: 1000
                      SaveFormat: 'Array'
                   MaxDataPoints: 1000
                         MaxStep: 'auto'
                         MinStep: 'auto'
          MaxConsecutiveMinStep: 1
                    OutputPoints: 'all'
                 OutputVariables: 'ty'
                          Refine: 1
                          RelTol: 1.0000e-003
                          Solver: 'ode45'
                    SrcWorkspace: 'base'
                           Trace: ''
                       ZeroCross: 'on'
                   SignalLogging: 'on'
               SignalLoggingName: 'logsout'
              ExtrapolationOrder: 4
         NumberNewtonIterations: 1
                         TimeOut: []
ConcurrencyResolvingToFileSuffix: []
         ReturnWorkspaceOutputs: []
   RapidAcceleratorUpToDateCheck: []
  RapidAcceleratorParameterSets: []
```

（3）simset 命令

通过 simget 命令，了解了仿真模型结构体参数的基本含义。那么在进行 Simulink 命令行仿真时，根据用户不同需求，来设置相关的参数进行系统仿真。simset 命令常用的调用格式如下：

```
options = simset(param, value, ...);
options = simset(old_opstruct, param, value, ...);
options = simset(old_opstruct, new_opstruct);
simset
```

从以上 4 条命令格式看出，除了第 4 条格式不返回任何值外，其他 3 条指令返回的均是

仿真模型仿真参数属性的结构体。

```
>>myopts = simset('MaxDataPoints', 100, 'Refine', 2);
[t,x,y] = sim('vdp', 10, myopts);
sim('vdp', 10, simset(simget('vdp'), 'SignalLogging', 'on'))
```

（4）simplot 命令

simplot 命令可以将仿真模型输出结果以示波器的形式绘制出来。其调用格式如下：

```
simplot(data);
simplot(time, data);
simplot(data, ports);
simplot(data, 'diff')
simplot(time, data, ports, 'diff')
```

采用如图 4-58 所示的仿真模型，实现如图 4-59 所示的仿真结果的命令代码如下：

```
>> vdp
set_param(gcs, 'SaveOutput', 'on')
set_param(gcs, 'SaveFormat', 'StructureWithTime')
sim(gcs)
simplot(yout)
```

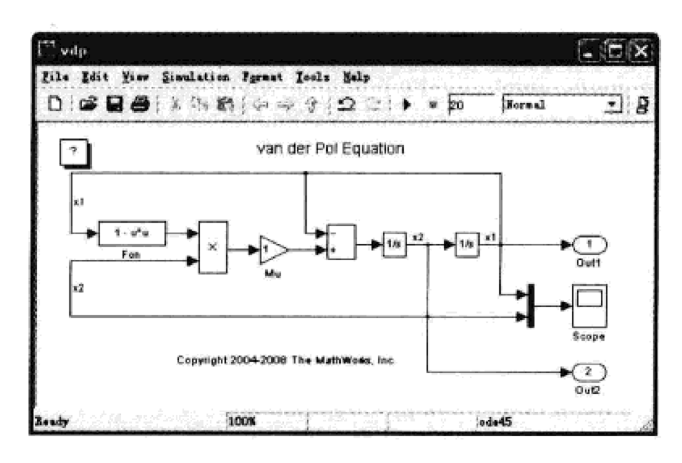


图 4-58　vdp 仿真模型

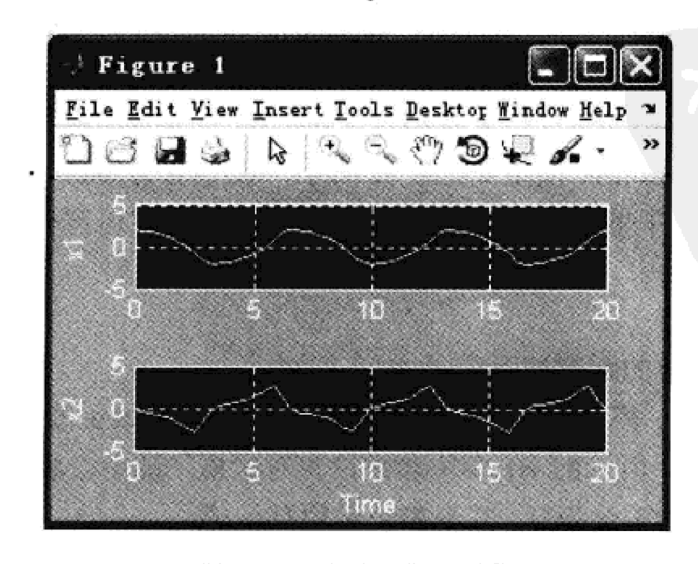


图 4-59　命令行仿真结果



第5章　Simulink 在控制系统中的应用

可用微分方程描述的系统称为连续时间系统。连续时间系统分为线性系统和非线性系统，而线性系统又分为线性定常（时不变）连续系统和线性时变连续系统。Simulink 模块库中用于连续时间系统建模的主要是 Continuous 模块组、Discontinuities 模块组和 Math Operations 模块组。

5.1　连续时间系统建模与仿真分析

5.1.1　线性连续时间系统

通常，线性连续时间系统的数学模型主要有：包含传递函数的结构图和包含状态空间表达式在内的微分方程。相应地，根据数学模型的不同，线性连续时间系统的模型可采用传递函数模块、积分模块或状态方程模块等来构建。

1．结构图数学模型

【例 5-1】 三阶控制系统结构如图 5-1 所示，建立系统的 Simulink 模型，并运行模型。

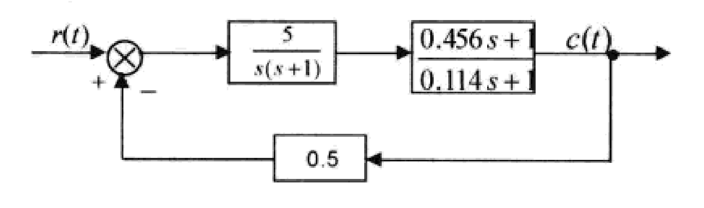


图 5-1　控制系统结构

本例的数学模型是控制系统的结构图，这是应用 Simulink 建模时，最简单、最方便、最直观的一种数学模型。

（1）构建 Simulink 模型

由图 5-1 构建的 Simulink 模型如图 5-2 所示，模型名为 xiu5_1.mdl。图中所需模块可分别在 Simulink 模块库中的信源模块组、连续模块组、数学运算模块组，以及信宿模块组中获得。

（2）模块参数的配置

图 5-2 中各模块参数的配置如下：

1）$r(t)$模块（即 Step 模块）：首先将模块名称由原来的 Step 改为 $r(t)$。再用鼠标双击该模块，即可打开如图 5-3 所示的 $r(t)$模块参数设置对话框。图中，将 Step time（阶跃信号发生时刻）文本框中默认的 1 改为 0，其余参数采用默认值。

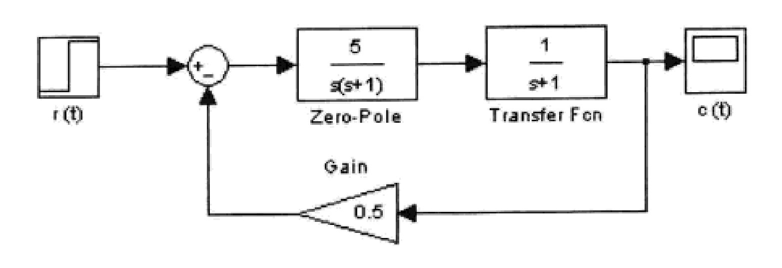


图 5-2　例 5-1 的 Simulink 模型

2）Sum 模块：用鼠标双击该模块，打开其参数设置对话框，将“List of signs”文本框中默认的“++”改为“+-”（系统为负反馈连接），如图 5-4 所示。



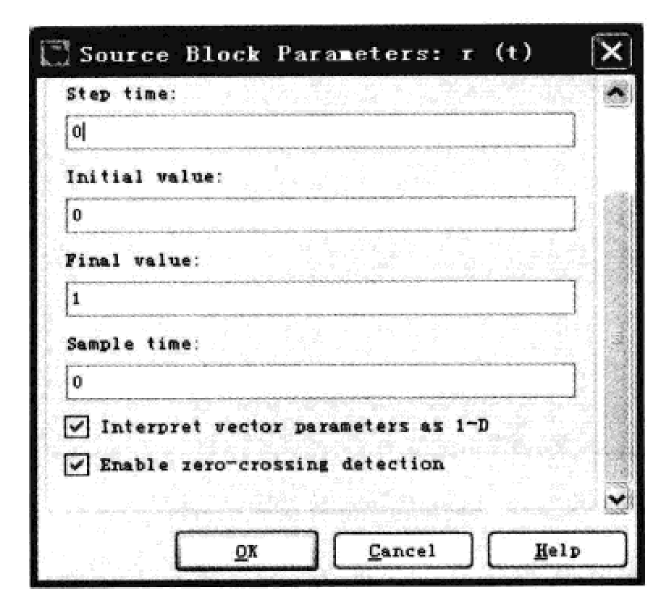


图 5-3　$r(t)$模块参数设置对话框

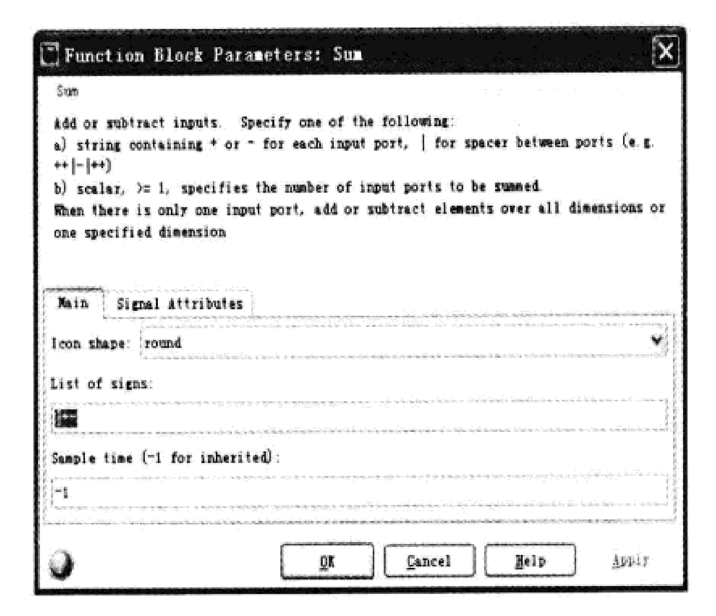


图 5-4　Sum 模块参数设置设置对话框

3）Zero-Pole 模块：用鼠标双击该模块，打开其参数设置对话框，分别在 Zeros、Ploes 和 Gain 文本框中填写传递函数的零点向量“[ ]”、极点向量“[0　-1]”和增益“[5]”，如图 5-5 所示。与此同时，该模块的图标也将显示新的传递函数。注意，由于此模块实现的零极点增益模型没有零点，因而在“Zeros”文本框填写空矩阵“[ ]”。

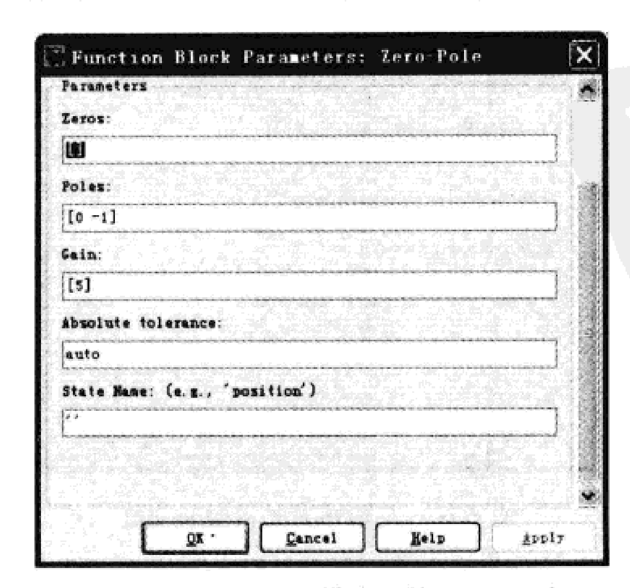


图 5-5　零极点增益模块参数设置对话框

4）Transfer Fun 模块：用鼠标双击该模块，打开其参数设置对话框，在“Numerator coefficient”文本框中填写分子多项式系数向量[0.456 1]，在“Denominator coefficient”文本框中填写分母多项式系数向量[0.114 1]，如图 5-6 所示。与此同时，该模块的图标也将显示新的传递函数。传递函数分子、分母多项式系数均按 s 降幂排列。

5）Gain 模块：首先选择模型窗口菜单“Format”→“Rotate Block”命令，旋转 Gain 模块的方向；然后，用鼠标双击 Gain 模块，打开其参数设置对话框，在“Gain”文本框中输入 0.5，如图 5-7 所示。

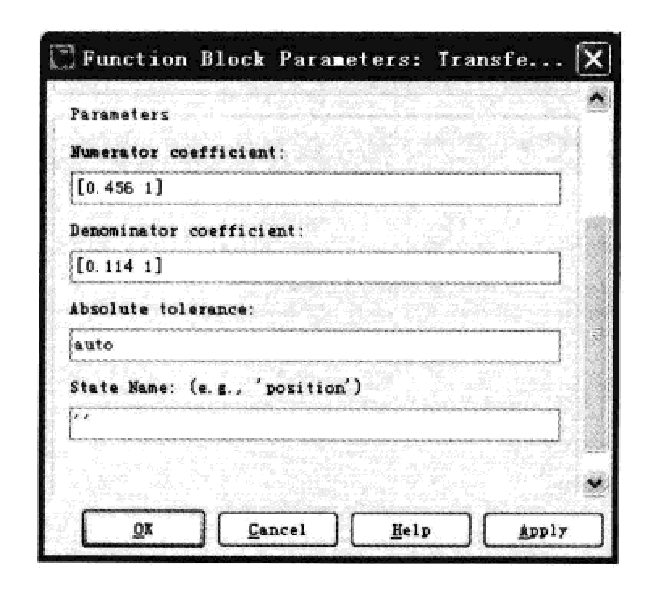


图 5-6 传递函数模块参数设置对话框

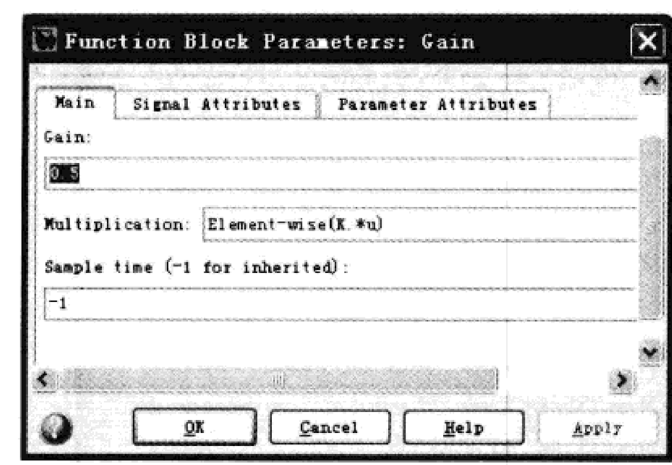


图 5-7 Gain 模块参数设置对话框

6）$c(t)$模块（即 Scope 模块）：首先将模块名称由原来的 Scope 改为 $c(t)$；然后，用鼠标双击该模块，出现示波器窗口；再用鼠标单击示波器窗口工具栏中的图标，打开如图 5-8 所示的示波器参数设置对话框；在“Data History”选项卡中，选中“Save data to workspace”复选框，这将使送入示波器的数据同时被保存在 MATLAB 工作空间默认名为 ScopeData 的时间结构体数组中。

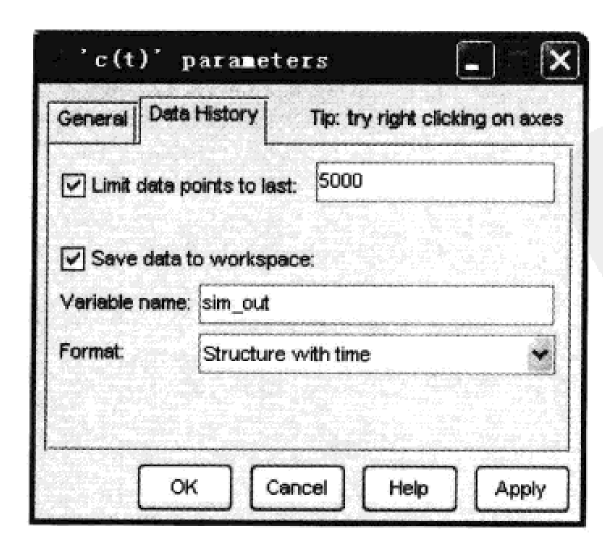


图 5-8 $c(t)$模块参数设置对话框

7）模型窗口 xiu5_1.mdl：仿真参数配置窗口中的各选项均采用默认值。

实际上，在输入参数之前，Simulink 中相应的模块都给出了较实用的提示，用户可以通过自己摸索的方式来学习每一个模块的使用方法。

（3）仿真运行



首先用鼠标双击 $c(t)$模块，打开示波器窗口；再用鼠标单击模型窗口“仿真启动”按钮▶，就可以看到示波器窗口中的 $c(t)$的变化曲线；还可再用鼠标单击显示屏幕上的“自动刻度”按钮，使得波形充满整个坐标框，仿真结果如图 5-9 所示。

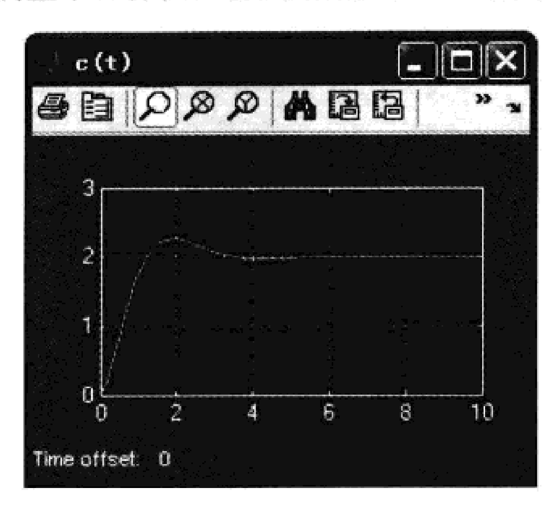


图 5-9 例 5-1 的仿真结果

（4）保存在 MATLAB 工作空间中的仿真数据的应用

本例通过示波器模块向工作空间存放了时间结构体数组 ScopeData。这组数据可独立地供用户作进一步分析时使用。下面的 MATLAB 程序（程序名为 li5_1.m）就说明了如何利用保存在 MATLAB 工作空间中的仿真数据（即示波器数据）ScopeData 绘制出所需的图形。

```
>> tt=ScopeData.time;                    %将时间结构体域的时间数据赋给 tt
xx=ScopeData.signals.values;             %将时间结构体域的数值数据赋给 xx
plot(tt,xx,'r:','LineWidth',2.5);        %绘制曲线
xlabel('t');
ylabel('c(t)')                           %为坐标轴添加说明
```

运行程序，效果如图 5-10 所示。

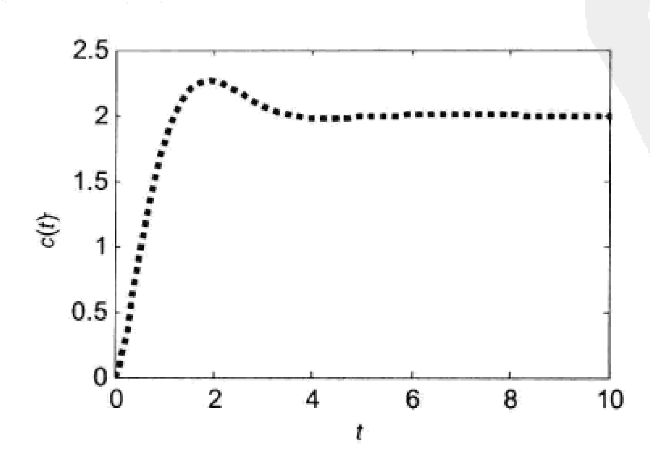


图 5-10 利用保存工作空间中的仿真数据绘制的曲线

2．微分方程数学模型

用微分方程描述的数学模型，可利用积分模块直接构建 Simulink 模型。

【例 5-2】 考虑如图 5-11 所示的强制阻尼二阶系统。图中，小车所受外力为 F，小车位移为 x。设小车质量 $m=10$，弹簧弹性系数 $k=3$，阻尼系数 $f=0.5$。设系统的初始状态为静止在平衡点处，即 $\dot{x}(0)=x(0)=0$，外力函数为幅值等于 1 的阶跃量。仿真此小车系统的运动。

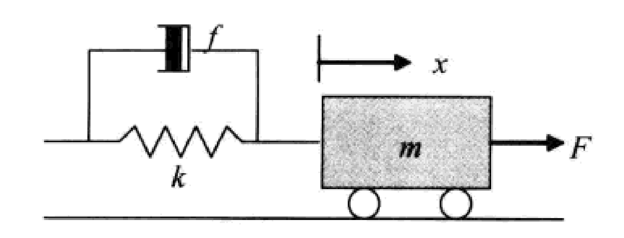


图 5-11 强制阻尼二阶系统

（1）建立系统数学模型

图 5-11 中，通过受力分析可知，有两个力影响着小车的运动：弹簧的弹力和阻尼器的阻尼力。弹性力为 kx，阻尼力为 $f\dot{x}$，小车的加速度为 $m\ddot{x}$。若忽略重力，这 3 个力的合力就为 F。根据牛顿第二定律，得到小车的运动方程为：

$$kx + f\dot{x} + m\ddot{x} = F \tag{5-1}$$

将 m,k,f 的值代入式（5-1），整理后得：

$$\ddot{x} + 0.1\dot{x} + 0.6x = 0.2F \tag{5-2}$$

将上述微分方程改写为：

$$\ddot{x} = u(t) - 0.1\dot{x} - 0.6x \tag{5-3}$$

式中，$u(t)=0.2F$。

（2）利用积分模块构建 Simulink 模型。

基于微分方程数学模型的仿真，实质上就是建立微分方程求解模型。因此，可利用积分模块采用逐次降阶积分法完成，即 $\ddot{x}$ 经积分模块作用输出 $\dot{x}$，$\dot{x}$ 再经积分模块作用就得到 x。而 $\dot{x}$ 与 x 经代数运算又产生 $\ddot{x}$。

依据上述思想，由式（5-3）所构建的 Simulink 模型如图 5-12 所示，模型名为 xiu5_2.mdl。图中，x'' 对应 $\ddot{x}$，x' 对应 $\dot{x}$。

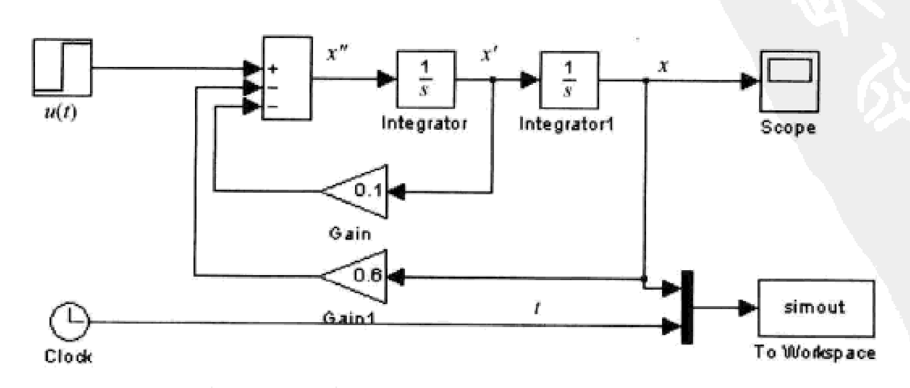


图 5-12 求解微分方程的 Simulink 模型



（3）模块参数的配置

图 5-12 中的模块参数配置如下：

1）$u(t)$模块：将模块名称由原来的 Step 改为 $u(t)$，在“Step time”文本框中输入 0，在“Final value”文本框中输入 0.1。

2）Gain 模块：在“Gain”文本框中输入 0.1。

3）Gain1 模块：在“Gain”文本框中输入 0.6。

4）Sum 模块：“Icon shape”（图标形状）下拉列表框中选择“rectangular”选项，使模块呈矩形；在“List of signs”文本框中输入“+--”。

5）Clock 模块：产生当前仿真时间数据 t，仅供 To Workspace 模块使用。

6）Mux 模块：“Number of inputs” 文本框中输入 2（默认值），如图 5-13 所示。该模块可将模型中的位移数据 x 与时间数据 t 组合成向量。

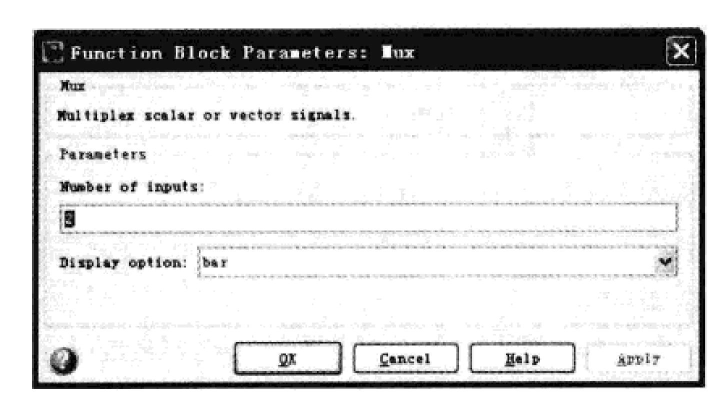


图 5-13 Mux 模块参数设置对话框

7）模型窗口 xiu5_2.mdl：将鼠标指针放置在模型窗口工具栏 10.0 内，将文本框内数值改为 50（即仿真结束时间），或单击模型窗口“Simulation”菜单下的“Configuration Parameters”命令，打开仿真参数配置对话框，在“Solver”选项组中的“Simulation time”选项中，将“Stop time”设置为 49。

（4）仿真运行

单击“Simulation”按钮，打开示波器窗口，就可在示波器窗口中显示出小车位移 x 随时间变化的轨迹，如图 5-14 所示。

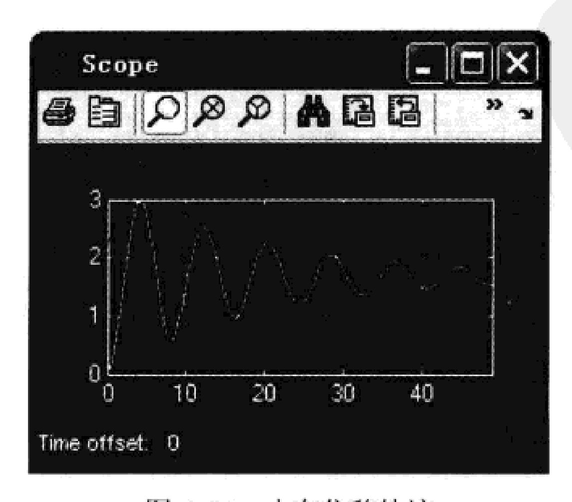


图 5-14 小车位移轨迹

（5）将数据保存到工作空间中

本例采用 To Workspace 模块以选定的矩阵方式向工作空间存放数组数据 simout。这组数据也可独立地供用户作进一步分析时使用。例如，在 MATLAB 命令窗口中输入：

```
>> x=simout(:,1);
t=simout(:,2);
plot(t,x)
```

3．积分模块的复位功能

积分模块的主要功能是构建诸如例 5-2 一类微分方程的 Simulink 模型。除此而外，利用积分器的复位功能还可以构建分段积分方程的 Simulink 模型。

【例 5-3】 构建如下积分方程的 Simulink 模型并求解。

$$f(t)=\begin{cases}\int_0^t 0.5t\mathrm{d}t & (0\leqslant t<5)\\ \int_0^t 0.5t\mathrm{d}t & (t\geqslant 5)\end{cases} \tag{5-4}$$

式中，$u(t)$ 是单位阶跃函数，初始条件为 $\dot{x}(0)=x(0)=0$。

本例说明如何产生带复位端口的积分模块及产生有两个显示窗口的示波器。

（1）构建积分方程求解模型

由式（5-4）构建的 Simulink 模型如图 5-15 所示，模型名为 xiu5_3.mdl。图中，积分模块与示波器模块均有两个输入端口，它们的产生方法如下：

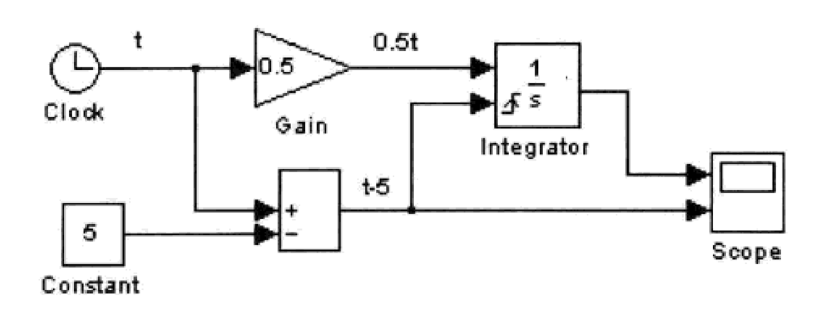


图 5-15 求解分段积分方程的 Simulink 模型

1）产生复位端口的积分模块。用鼠标双击 Inte grator 模块，打开其参数设置对话框；在 External reset（外复位）下拉列表框中选择“rising”选项；用鼠标单击“OK”按钮，积分模块就呈现如图 5-15 所示的两个端口，下端口为复位端口，该端口旁的符号表示此端口信号由负变正的瞬间，该积分器被强迫置为零。

2）产生有两个显示窗口的示波器。用鼠标双击 Scope 模块，打开示波器窗口，如图 5-16 所示；再用鼠标单击该窗口工具栏中的图标按钮，打开示波器属性对话框；在“Number of axes”文本框中输入 2，用鼠标单击“OK”按钮，就获得两端口示波器，同时出现图 5-17 所示的两个显示窗口。

（2）仿真模型参数配置

1）Clock 模块：生成时间变量 t。

2）Constant 模块：“Constant value”文本框中输入 5。

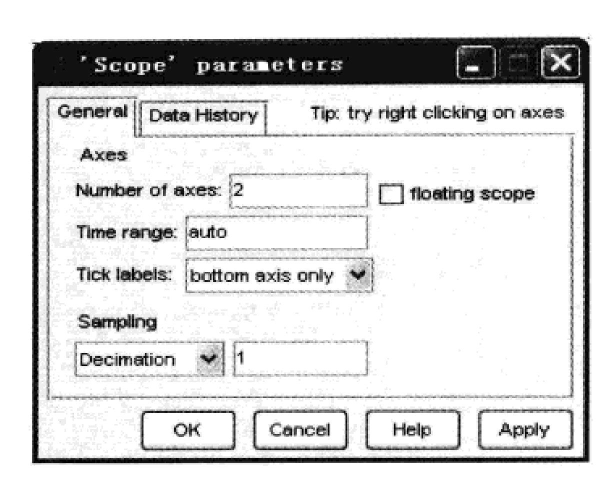




图 5-16 Scope 模块参数设置对话框

3）Sum 模块："List of sings"文本框中输入"+-"。

4）增益模块："Gain" 文本框中输入"0.5"。

5）示波器模块：自上而下，在示波器第一个显示窗口坐标框内单击鼠标右键，系统弹出一个现场菜单，选择"Axes properties..."命令，打开纵坐标设置对话框；在 Y-min 和 Y-max 文本框中分别输入 0 和 10（纵坐标下、上限），在"Title"文本框中输入 Simulink f(t)。示波器第二个显示窗的纵坐标下、上限采用默认值，"Title"文本框中输入 *t*-5。

6）模型窗口 xiu5_3.mdl：仿真参数配置窗口各选项均采用默认值。

（3）仿真运行

运行仿真模型，在示波器窗口中显示出 Simulink f(t)和 t-5 曲线，效果如图 5-17 所示。

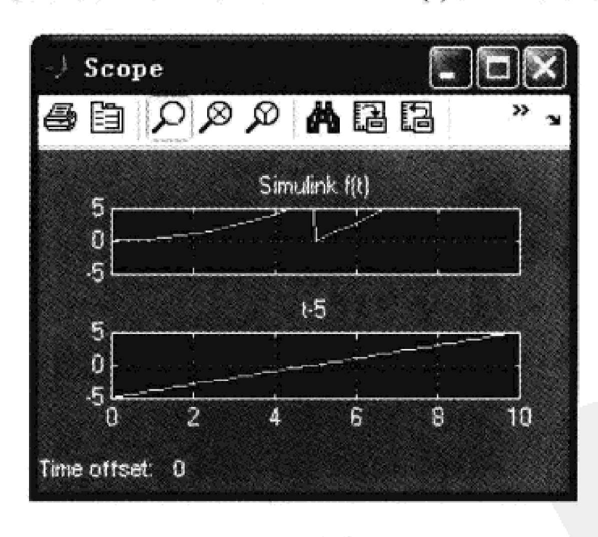


图 5-17 仿真效果

4. 单位脉冲函数的生成

像其他物理体系中不存在理想单位脉冲一样，Simulink 模块库中也没有现成的单位脉冲标准模块，但可以采用某种近似方法产生。

【例 5-4】 已知控制系统的状态方程为：

$$\dot{x}(t)=\begin{pmatrix}0 & 1\\ -0.4 & -0.2\end{pmatrix}x(t)+\begin{pmatrix}0\\ 0.2\end{pmatrix}u(t)$$

$$y(t)=[1 \quad 0]x(t)$$

试求系统的单位脉冲响应。

本例主要说明单位脉冲函数的生成方法及状态方程模块的使用。

（1）单位脉冲函数的数学含义及近似实现

单位脉冲函数在数学上定义为：

$$\delta(t)=\begin{cases}0 & (t=0)\\ \infty & (t\neq 0)\end{cases} \tag{5-6}$$

且满足

$$\int_{-\infty}^{\infty}\delta(t)=1 \tag{5-7}$$

近似构造单位脉冲函数的思路是：用一个面积为 1 的“窄高”脉冲近似，其数学表达式为：

$$\delta(t)=M\cdot 1(t)-M\cdot 1(t-d) \tag{5-8}$$

式中，$1(t)$ 为单位阶跃函数；M 为近似脉冲幅度；d 为近似脉冲宽度，且 $M\cdot d=1$。

说明：d 的选择要考虑下述两方面的因素：

1）脉冲宽度应远小于被研究系统的最快动态模式（系统特征根或特征值的实部绝对值的最大值）。

2）脉冲宽度不能太小，以免引起严重的圆整或截断误差。

本例系统的特征值可采用下述 MATLAB 命令求出：

```
>> eig([0 1;-0.4 -0.2])
```

运行程序，输出结果如下：

```
ans =
   -0.1000 + 0.6245i
   -0.1000 - 0.6245i
```

即，系统特征值为 $\lambda_{1,2}=-0.1\pm \mathrm{i}0.6245$。

由于系统特征值实部的绝对值为 0.1，因此取近似脉冲宽度 $d=0.01$，幅度 $M=100$，代入式（5-8），得

$$\delta(t)=100\cdot 1(t)-100\cdot 1(t-0.01) \tag{5-9}$$

（2）构建 Simulink 模型及参数配置

由式（5-9）构建的 Simulink 模型如图 5-18 所示，模型名为 xiu5_4.mdl。图中，各模块参数配置如下。

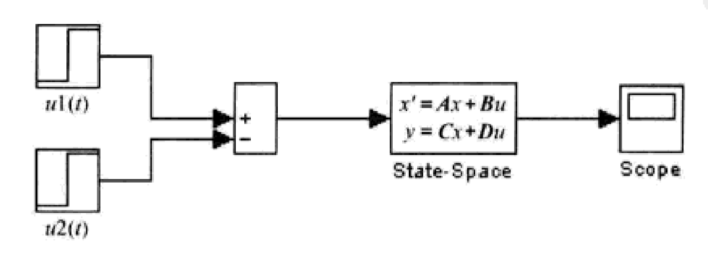


图 5-18　具有近似单位脉冲的 Simulink 模型

1）$u1(t)$模块："Step time"文本框中输入 0，"Final time"文本框中输入 88。

2）$u2(t)$模块："Step time"文本框中输入 0.01，"Final time"文本框中输入 88。

3）Sum 模块："List of Signs"文本框中输入+-。

4）State-Space 模块：在矩阵 ***A***、***B***、***C***、***D*** 栏中依次填写[0, 1; −0.4, −0.2]、[0; 0.2]、[1, 0]、0。



5）模型窗口 xiu5_3.mdl；将模型窗口工具栏图标 10.0 文本框内数值改为 20，即将仿真终止时间设置为 20。其余仿真参数采用默认值。

（3）仿真运行

打开示波器窗口，运行仿真，在示波器窗口显示出 $y(t)$曲线，如图 5-19 所示。

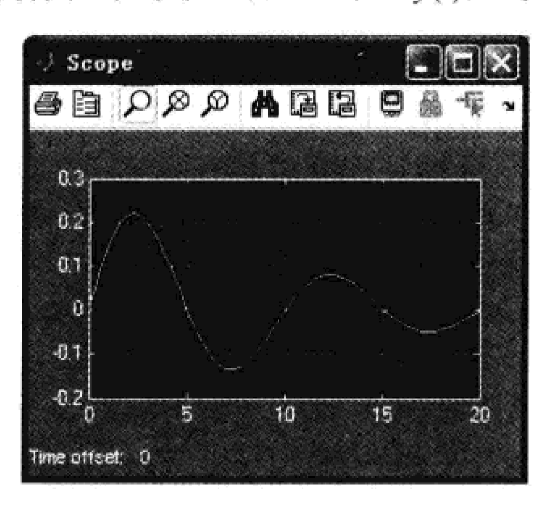


图 5-19 仿真效果

5.1.2 非线性连续时间系统

在工程中，严格意义上的线性系统很少存在，大量的系统或元器件都是非线性的。非线性系统的 Simulink 建模方法很灵活。

1．典型的非线性模块及其应用

为了提高仿真能力，Simulink 模块库中包含了许多典型非线性模块，如 Dead Zone 模块、Saturation 模块、Relay 模块及 Backlash 模块等。

应用 Simulink 构建非线性连续时间系统的仿真模型时，根据非线性元器件参数的取值，既可使用典型非线性模块直接实现，也可通过对典型非线性模块进行适当组合实现。当然，还可以采用 Fcn 函数模块或其他 Simulink 模块库中的模块实现。

【例 5-5】 设具有饱和非线性特性的控制系统如图 5-20 所示。通过仿真研究 $K=4$ 和 $K=16$ 时系统的运动。

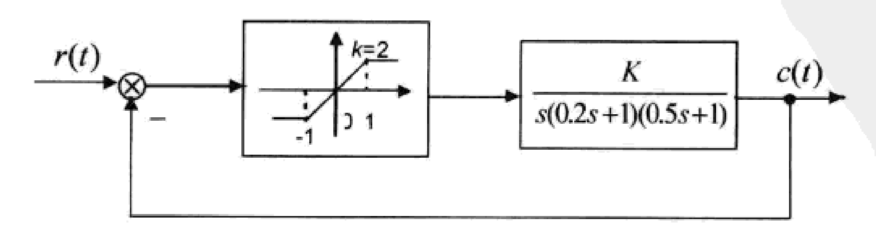


图 5-20 具有饱和非线性特性的控制系统的结构

（1）构建 Simulink 模型

由图 5-20 所构建的 Simulink 模型如图 5-21 所示，模型名为 xiu5_5.mdl。

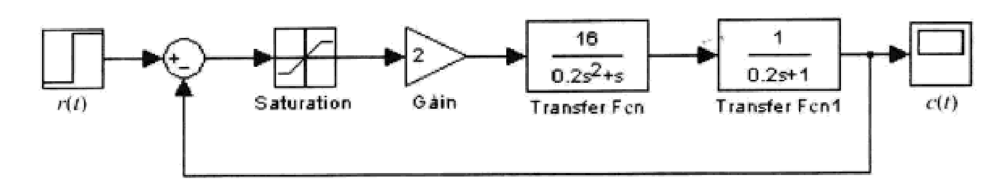


图 5-21　Simulink 模型

由于系统中的饱和非线性特性的线性段斜率 k=2，而 Simulink 模块库中的饱和非线性模块线性段斜率取值只能为 1。故图 5-20 中，在饱和非线性模块之后又串接了一个增益模块（增益为 2），以实现线性斜率为 2 的饱和非线性特性。

（2）仿真模型参数配置

图 5-20 中各模块参数配置如下：

1）$r(t)$模块："Step time"文本框中输入 0，"Final time"文本框中输入 1。

2）Sum 模块："List of signs"文本框中输入+-。

3）Saturation 模块："Upper limit"（饱和上限）文本框中输入 1；"Lower limit"（饱和下限）文本框中输入-1。

4）Gain 模块："Gain"文本框中输入 2。

5）G1(s)模块："Numerator"文本框中输入[16]，"Denominator"文本框中输入[0.2 1 0]。

6）G2(s)模块："Numerator"文本框中输入[1]，"Denominator"文本框中输入[0.5 1]。

7）模型窗口 xiu5_5.mdl：仿真参数配置窗口各选项均采用默认值。

（3）仿真运行

打开示波器窗口，运行仿真，则得到 K=15 时系统的响应曲线，如图 5-22 所示。显见，此时非线性系统的运动出现自激振荡。

进一步，将传递函数 $G1(s)$模块的"Numerator"文本框设置由[16]改为[4]，其余参数不变。同样可以得到 K=15 时非线性系统的响应曲线，如图 5-23 所示。此时非线性系统的运动已经没有自激振荡了。

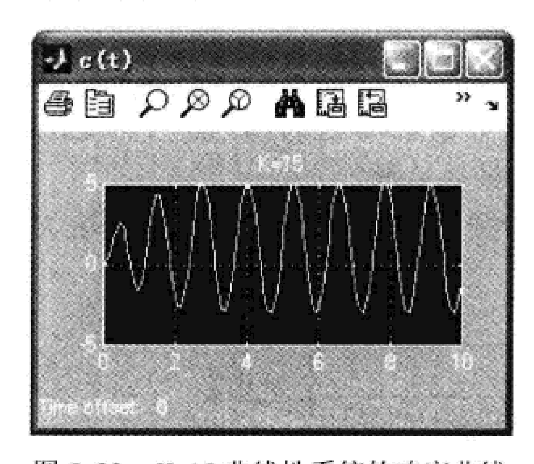


图 5-22　K=15 非线性系统的响应曲线

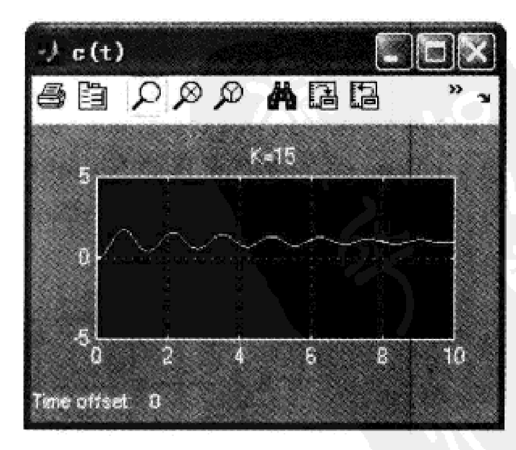


图 5-23　K=15 非线性系统的响应曲线

2. 任意函数模块及其应用

在 Simulink 模块库中，除间歇、死区、饱和等函数形式固定的模块外，还有若干个函数

形式可由用户根据需要定义的“任意函数”模块，主要有：Fcn 模块（函数组合模块）、MATLAB Fcn 模块（MATLAB 函数模块）和 Look-up Table 模块等，其模块图如图 5-24 所示。



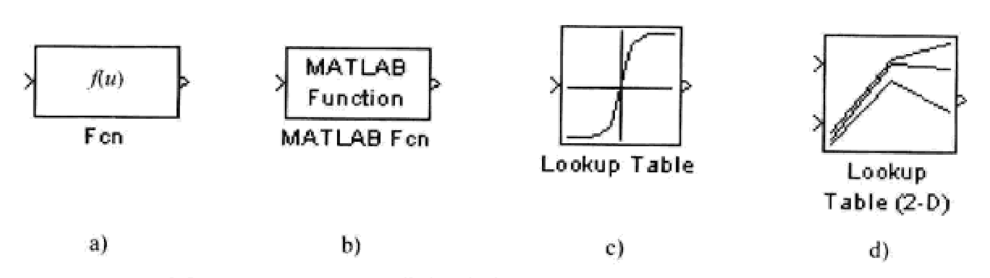


图 5-24 Simulink 模块库中的“任意函数”模块图标

a) Fcn 模块 b) MATLAB Fcn 模块 c) 一维 Lookup Table 模块 d) 二维 Lookup Table 模块

（1）Fcn 模块

Fcn 模块位于 User Define Function（用户自定义）模块组中，模块图标如图 5-24a 所示，其参数设置对话框如图 5-25 所示。图中，Expression（表达式）文本框必须填写函数表达式（即函数的解析式），且必须遵循下述规则：

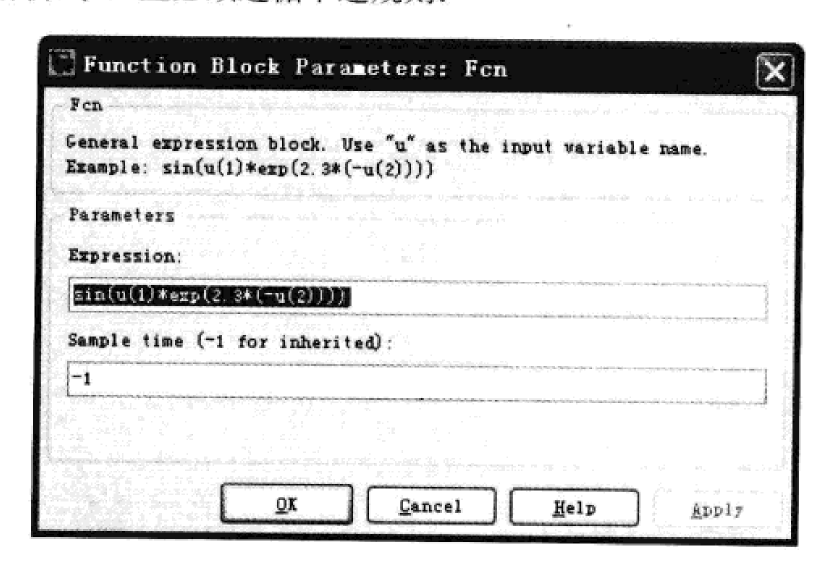


图 5-25 Fcn 模块参数设置对话框

1）模块的输入可以是标量或向量，但输出一定是标量。模块输入是标量时，必须用 u 作为变量名；输入为向量时，必须用 $u(1)$，$u(2)$ 等向量作为元素名。图 5-25 中“Expression”文本框的内容为默认表达式。

2）表达式符合 C 语言格式，执行的是标量运算，计算结果就是模块的输出。

3）表达式中引用的其他标量形式的参量必须存在于 MATLAB 工作空间中。

2）MATLAB Fcn 模块

MATLAB Fcn 模块也位于用户自定义模块组中，模块图标如图 5-24b 所示，其参数设置对话框如图 5-26 所示。图中，“MATLAB function”文中框中填写表达式或函数文件名，且应遵循下述规则：

1）模块的输入、输出都可以是标量或向量。

2）表达式的书写规则与 Fcn 模块相同；函数编写符合 M 函数文件基本结构及规则。

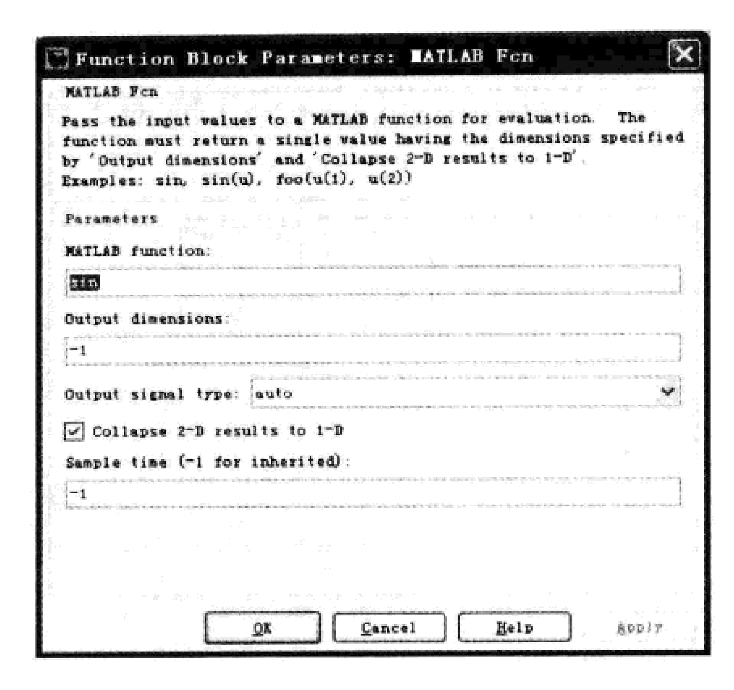

Function Block Parameters: MATLAB Fcn
MATLAB Fcn
Pass the input values to a MATLAB function for evaluation. The function must return a single value having the dimensions specified by 'Output dimensions' and 'Collapse 2-D results to 1-D'.
Examples: sin, sin(u), foo(u(1), u(2))
Parameters
MATLAB function:
sin
Output dimensions:
-1
Output signal type: auto
Collapse 2-D results to 1-D
Sample time (-1 for inherited):
-1
OK Cancel Help Apply

图 5-26 MATLAB Fcn 函数模块参数设置对话框

3）表达式或函数的输出必须与该模块的输出维数匹配，否则就会出现错误。

该模块可以进行的运算比 Fcn 模块复杂，但速度较慢。

（3）Lookup Table 模块

Lookup Table 模块位于 Lookup Tables（查表）模块组中，有一维、二维及 N 维之分。图 5-24c 是一维 Look-up Table 模块图标，而图 5-24d 则是二维 Lookup Table 模块图标。此类模块可根据所给表格对输入进行“插补”或“外推”运算。

【例 5-6】 将图 5-20 所示非线性控制系统中的饱和非线性用 MATLAB Fcn 函数模块实现。

1）构建 Simulink 模型。由图 5-20 所构建的 Simulink 模型如图 5-27 所示，模型名为 xiu5_6.mdl。

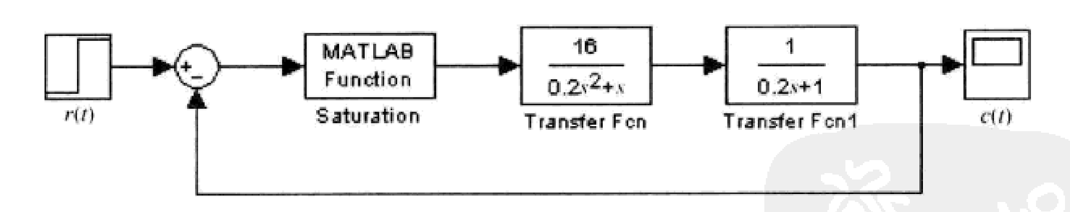


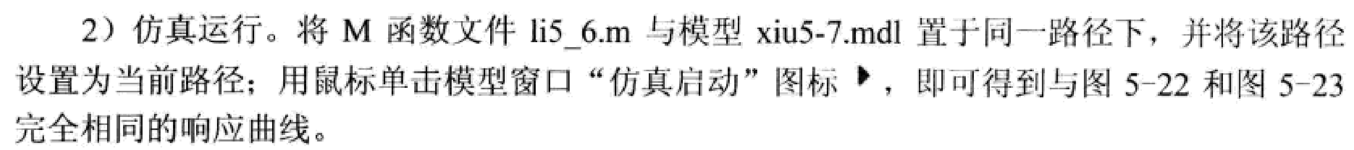
图 5-27 具有 MATLAB Fcn 模块的非线性系统仿真模型

图中，Saturation 模块（即 MATLAB Fcn 模块）实现饱和非线性特性，函数名为 li5_6.m。因此，在该模块参数设置对话框（见图 5-26）的“MATLAB function”文本框中填写函数名为 bh（可省略扩展名），其 M 函数文件如下：

```
function y= li5_6(u)
  if abs(u)<=1
          y=2*u;
  elseif u>1
```

```
            y=2;
      else y=-2;
      end
    end
```





2）仿真运行。将 M 函数文件 li5_6.m 与模型 xiu5-7.mdl 置于同一路径下，并将该路径设置为当前路径；用鼠标单击模型窗口“仿真启动”图标 ▶，即可得到与图 5-22 和图 5-23 完全相同的响应曲线。

【例 5-7】 蹦极跳是一种挑战身体极限的运动，蹦极者系着一根弹力绳从高处的桥梁向下跳。在下落的过程中，蹦极者几乎处于失重状态。试应用 Simulink 对蹦极跳系统进行仿真研究。

1）蹦极跳系统数学模型。按照牛顿运动规律，自由下落物体的位置由式（5-10）确定：

$$m\ddot{x} = mg - a_1\dot{x} - a_2\,|\,\dot{x}\,|\,\dot{x} \tag{5-10}$$

式中，m 为物体的质量；g 为重力加速度；x 为物体的位置，第二项与第三项表示空气的阻力；a_1、a_2 为空气阻力系数。

若选择桥梁作为蹦极者开始跳下的起点，即 $x=0$，表明位置 x 的基准为蹦极者开始跳下的位置，并设低于桥梁的位置为正值，高于桥梁的位置为负值。

如果蹦极者系在一个弹性常数为 k 的弹力绳索上，定义绳索下端的初始位置为 0，则其对落体位置的影响力：

$$bx = \begin{cases} -kx & (x > 0) \\ 0 & (x \leqslant 0) \end{cases} \tag{5-11}$$

这样，整个蹦极跳系统的数学描述为

$$m\ddot{x} = mg + bx - a_1\dot{x} - a_2\,|\,\dot{x}\,|\,\dot{x} \tag{5-12}$$

显见，蹦极跳系统是一个典型的非线性连续时间系统。

2）蹦极跳系统仿真问题描述。假设：桥梁距离地面为 60m；蹦极者的起始位置为绳索的长度-30m，即 $x(0) = -30\text{m}$；蹦极者起始速度为零，即 $\dot{x}(0) = 0$；其余参数分别为：

$k = 25$，$a_1 = a_2 = 1$，$m = 60\text{kg}$，$g = 10\text{m}/s^2$。

要求：通过仿真，分析此蹦极跳系统对体重为 60kg 的蹦极者而言是否安全。

3）蹦极跳系统 Simulink 模型及参数配置。由式（5-11）和式（5-12）可构建蹦极跳系统的 Simulink 模型，如图 5-28 所示，模型名为 xiu5_7.mdl。

图中主要模块的参数配置如下：

- C1 模块（即 Constant 模块）：“Constant value”文本框中输入 60*10。
- C2 模块：“Constant value”文本框中输入 60。
- Integrator1 模块：“Initial condition”文本框为默认值 0。
- Integrator 模块：“Initial condition”文本框为默认值-30。

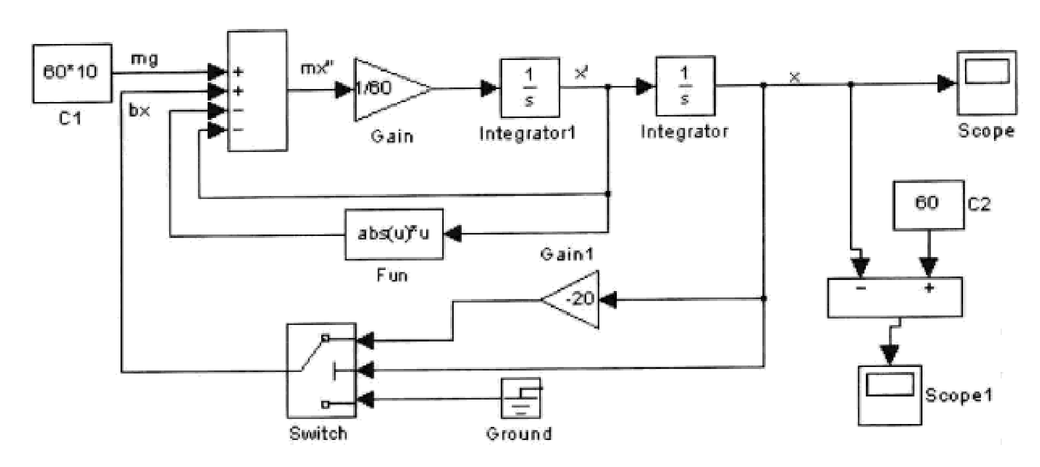


图 5-28　蹦极跳系统的 Simulink 模型

4）Gain 模块：“Gain”文本框中输入 1/60。

5）Gain 模块：“Gain”文本框中输入-20（即绳索弹性常数 k 的负值）。

6）Fun 模块：“Expression”文本框中输入 abs(u)*u。

7）Switch 模块：位于 Signal Routing 模块组中。该模块为两个输入选项模块，其功能是根据第二个输入决定输出其他两个输入中的哪一个：若第二个输入大于或等于参数 Threshold 的值，则输出第一个输入；否则，输出第三个输入。其功能示意如图 5-29 所示。

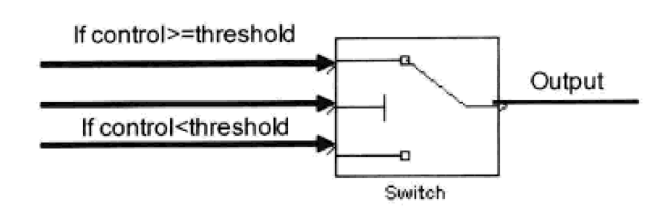


图 5-29　Switch 模块功能

9）Scope 模块：显示蹦极者的相对位置，即蹦极者相对于桥梁的位置。

10）Scope1 模块：显示蹦极者的绝对位置，即蹦极者相对于地面的距离。

11）模型窗口 xiu5_7.mdl：将仿真结束时间设置为 100，其余仿真参数均采用默认值。

（4）仿真运行

打开 Scope1 示波器窗口。运行仿真，则得到 k=25 时系统的响应曲线，如图 5-30 所示，图中显示的是蹦极者相对于地面的距离。

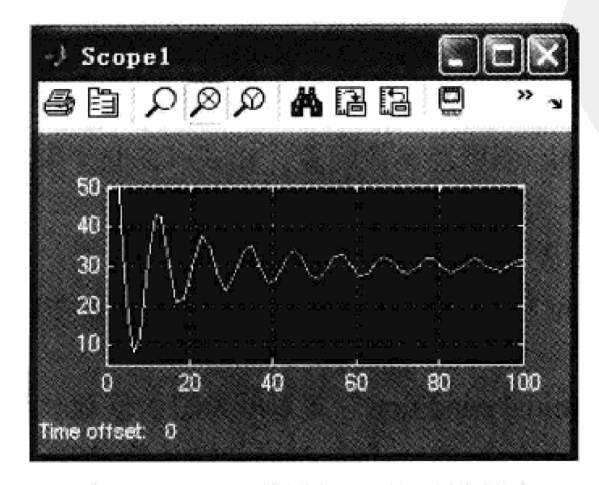


图 5-30　k=25 蹦极相对于地面的距离

（5）仿真结果分析

由图 5-30 的仿真结果知，对于体重 60kg 的蹦极者来说，此系统是不完全的。因为蹦极者与地面之间的距离出现了负值，即蹦极者在下落的过程中会触地，而完全的蹦极系统要求两者之间的距离应该大于 0。



若将弹力绳索的弹性常数 k 增大，上述情况就会改变。图 5-31 为 k=30 时的仿真结果。显见，蹦极者与地面之间的距离为正值。

因此，必须使用弹性常数较大的弹性绳索，才能保证蹦极者的安全。当然，在蹦极者触地的情况下，系统的动态方程会发生改变，系统输出结果也将发生变化。上述蹦极跳系统的仿真结果并没有考虑这一点，即假设蹦极者距离地面足够远，不会触发。

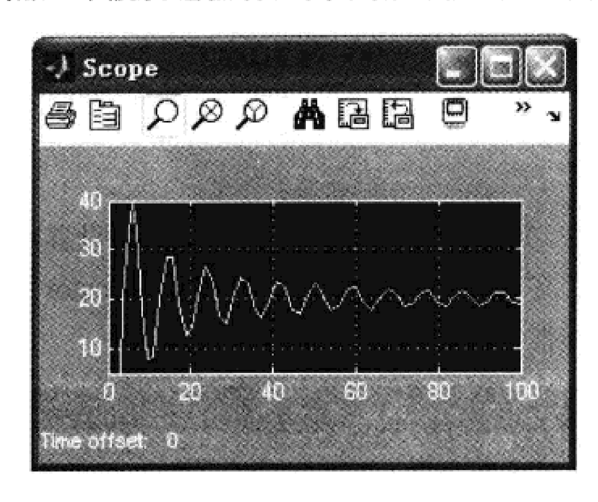


图 5-31 k=30 蹦极相对于地面的距离

5.2 离散系统建模与仿真分析

5.2.1 离散时间系统建模介绍

离散系统包括离散时间系统和连续—离散混合系统。离散时间系统既可以用差分方程描述，也可以用脉冲传递函数描述（对于线性定常离散时间系统）。而连续—离散混合系统则可用微分—差分方程描述，或用传递函数—脉冲传递函数描述（对于线性定常连续—离散混合系统）。

在 Simulink 模块库中，除有一个专门的 Discrete 模块组外，其他一些模块组，如数学运算模块组、信宿模块组、信源模块组中的几乎所有模块也都能用于离散系统建模。

采样周期是所有离散模块最重要的参数。在所有离散模块的参数设置对话框里，在 Sample time（采样周期）文本框中可以填写标量 T_s 或二维向量[T_s, offset]。这里，T_s 是指定的采样周期；offset 是时间偏移量，它可正可负，但绝对值总小于 T_s。实际的采样时刻 $t = T_s + \text{offset}$。

对于纯离散系统，优先使用 Solver 解算器算法是 discrete，但该算法完全不能处理连续时间系统。其他解算器算法都同时适用于离散时间系统和连续时间系统。

通常，离散系统仿真建模最常使用的是 discrete 模块组中的模块。

5.2.2 定常离散时间系统建模与仿真

【例 5-8】 线性定常离散控制系统 Simulink 模型如图 5-32 所示，并保存为 xiu5_8.mdl。已知 $r(t)=1(t)$，试根据 Simulink 仿真模型，求系统周期 $T=0.1\text{s}$ 和 $T=1\text{s}$ 时系统的单位阶跃响应。

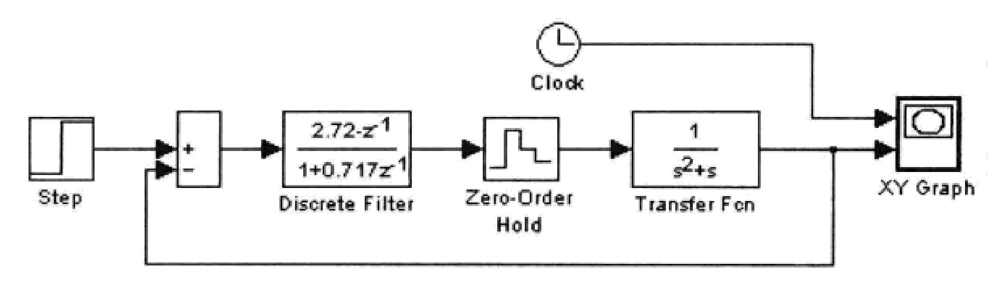


图 5-32 Simulink 模型

（1）仿真参数配置

1）Discrete Filter 模块：实现数字控制器 $D(z)$，其参数设置对话框如图 5-33 所示。图中，“Numerator coefficient”文本框中输入[2.72-1]，Denominator coefficient 文本框中输入[1 0.717]，“Sample time” 文本框中输入 1。

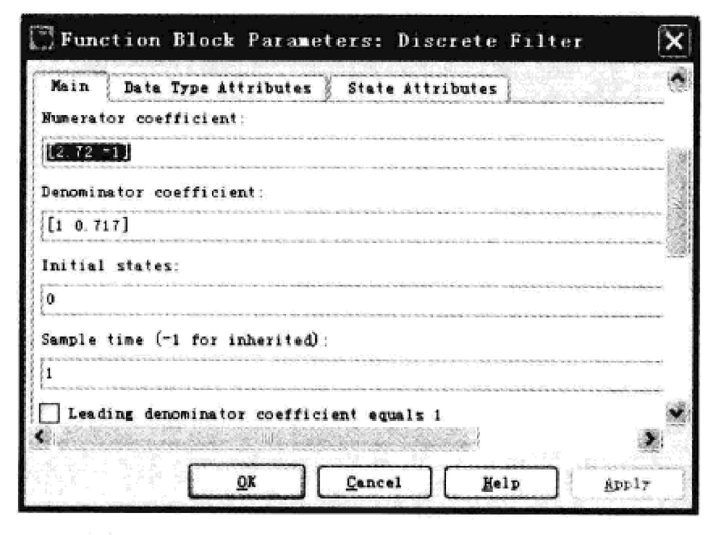


图 5-33 Discrete Filter 模块参数设置对话框

2）Zero-Order Hold 模块：在其参数设置对话框的“Sample time”文本框中输入 0.1，效果如图 5-34 所示。

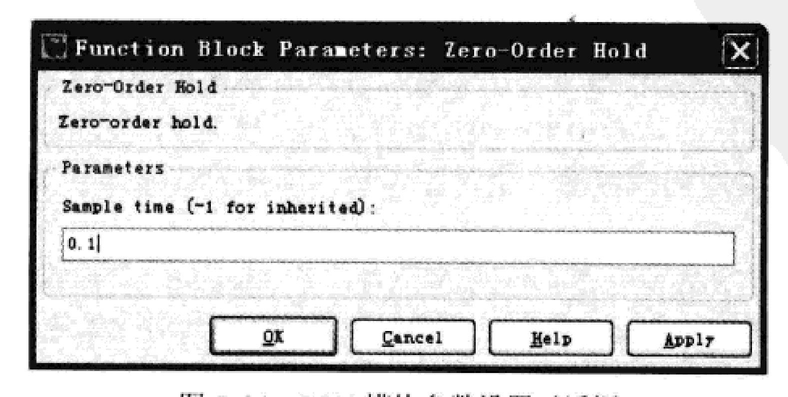


图 5-34 ZOH 模块参数设置对话框

3）Clock 模块：产生时间 t，与 XY Graph 模块配合使用。

4）XY Graph 模块：双击该模块，打开如图 5-35 所示的参数设置对话框；"x-min" 文本框中输入（x 坐标下限）0，"x-max" 文本框中输入（x 坐标上限）15；"y-min"（y 坐标下限）文本框中输入 0，y-max（y 坐标上限）文本框中输入 1.5。



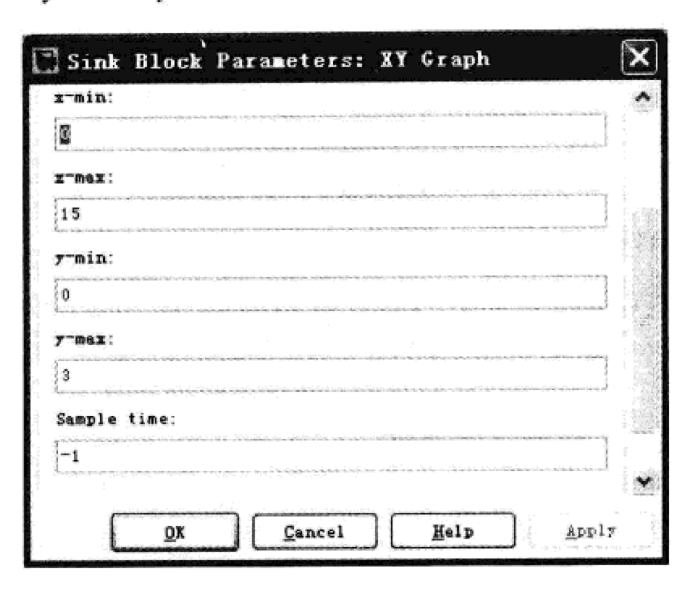


图 5-35　XY Graph 模块参数设置对话框

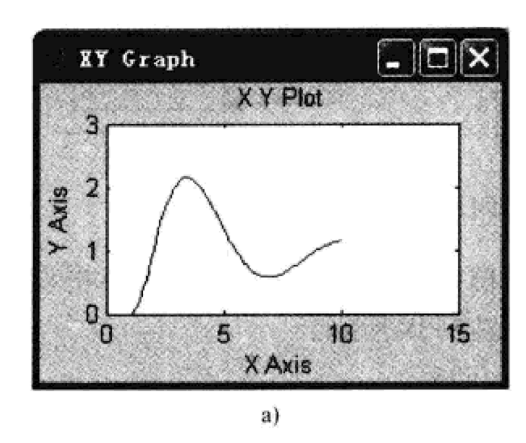
注意： XY Graph 模块与 Scope 模块都可以将仿真运行数据生成二维曲线。但前者可以任意两组数据作为二维曲线的横坐标（或纵坐标），因而具有很大的灵活性；而后者只能以仿真运行时间 t 作为横坐标。

5）模型窗口 xiu5_8.mdl：将仿真结束时间设置为 15，采用变步长解算器，仿真算法选择 ode45。

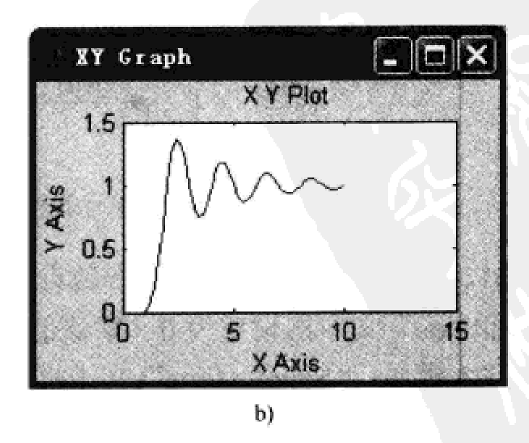
（2）仿真运行

单击模型窗口运行仿真，同时 XY Graph 模块自动打开，并实时显示输出响应曲线，如图 5-36a 所示。

进一步，将 Discrete Filter 模块和 ZOH 模块参数设置对话框中的"Sample time"文本框参数设置为 1，并运行仿真，仿真结果如图 5-36b 所示。



a)



b)

图 5-36　系统的单位阶跃响应曲线

a) T=0.1　b) T=1

显然，采样周期 T=0.1s 时系统的动态性能优于采样周期 T=1s，说明采样周期的大小对系统性能有很大影响。

5.2.3 非线性离散时间系统建模与仿真

【例 5-9】 离散控制系统比例控制器设置示例。已知，被控对象的离散状态空间表达式为：

$$\begin{cases} x_1(k+1)=x_1(k)+0.1x_2(k) \\ x_2(k+1)=-0.05\sin x_1(k)+0.094x_2(k)+u(k) \\ y(k)=x(k) \end{cases} \tag{5-13}$$

式中，$x_1(k)$ 和 $x_2(k)$ 为状态变量；$u(k)$ 为被控对象的输入；$y(k)$ 为受控对象的输出，该受控过程的采样周期为 0.1s。要求，应用采样周期为 0.25s 的比例控制器，输出显示的采样周期为 0.5s。

本例是一个多速率非线性离散时间系统。

（1）比例控制器数学模型

比例控制器工作原理：根据期望输出 $y_c(k)$ 和实际输出 $y(k)$ 之差产生控制输入 $u(k)$，其数学模型为：

$$u(k)=K_p[y_c(k)-y(k)] \tag{5-14}$$

式中，K_p 为比例系数，本例可取 K_p=1。

（2）仿真模型及参数配置

由式（5-13）和式（5-14）可构建系统的 Simulink 模型，如图 5-37 所示，模型名为 xiu5_9.mdl。图中的主要模块参数配置如下：

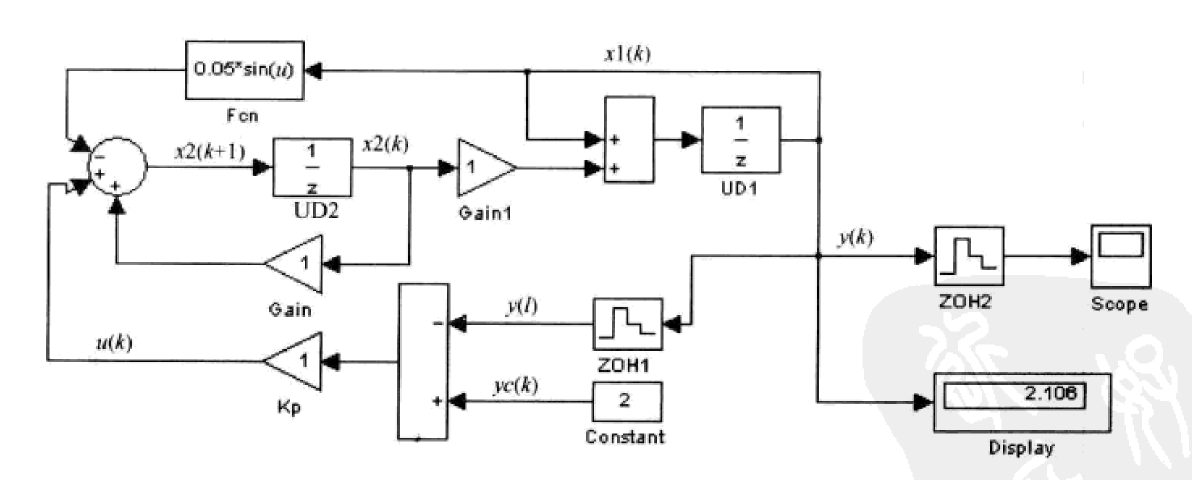


图 5-37 多速率非线性离散时间系统 Simulink 模型

1）UD1、UD2 模块：分别在其参数设置对话框中将“Sample time”文本框设置为 0.1，“Initial conditions”文本框设置为 0（默认值），如图 5-38 所示。

2）ZOH1 模块：“Sample time”文本框中输入 0.25。

3）ZOH2 模块：“Sample time”文本框中输入 0.5。

4）Command 模块：设置比例控制器的比例系数。“Contant value”文本框中输入 2。

5）Scope 模块：显示输入 y 的历史记录。

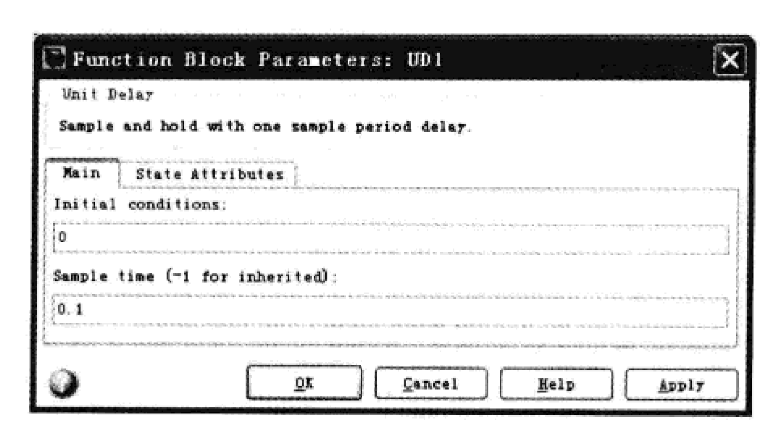


图 5-38　UD1 模块参数设置对话框



6）模型窗口 xiu5_9.mdl：仿真参数全部采用默认配置。同时，单击“Format”菜单下的“Port”子菜单下的“Signal Displays”命令项下的“Sample time colors”命令，则模型中不同采样周期的模块和连线以不同颜色表示。本例中，采样速度最快的受控过程部分会显示为红色；速度次多的控制器部分则显示为绿色；而蓝色显示的则是 y 的历史记录部分。

（3）仿真运行

将仿真结束时间设置为 10s。Display 模块在仿真过程中实时地显示 $y(k)$的数值。图 5-37 中 Display 模块上的数值是仿真结束时的 $y(k)$值。

5.3　控制系统设计分析与示例

5.3.1　简单闭环控制系统的仿真分析

在此节中，将对简单的闭环控制的调速系统进行 PI 校正设计，并验算设计后系统的时域与频域性能指标是否满足要求。

【例 5-10】 已知晶闸管-直流电动机单闭环系统（V-M 系统）的 Simulink 动态结构如图 5-39 所示。

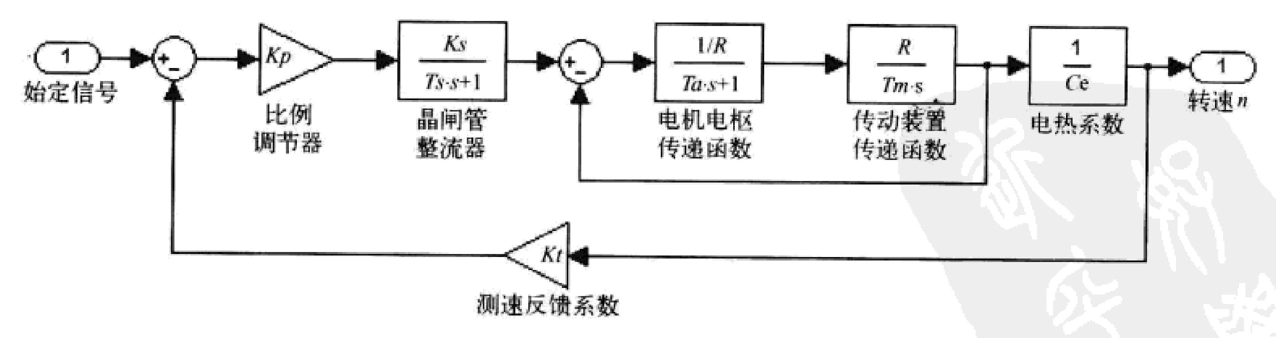


图 5-39　转速单闭环调速系统的 Simulink 动态结构

图 5-39 中，电动机参数：$P_N = 2.2\text{kW}$，$n_N == 1500\text{r/min}$，$U_N = 220\text{V}$，$I_N = 12.5\text{A}$，电动机电枢电阻 $R_a = 1\Omega$，V-M 系统主电路总电阻 $R = 2.9\Omega$，电枢主回路总电感 $L = 40\text{mH}$，拖动系统运动部分飞轮矩 $GD^2 = 1.5\text{N}\cdot\text{m}^2$，整流触发装置的放大系数 $K_s = 44$，三相桥平均失控时间 $K_s = 0.00167\text{s}$。①要求系统调速范围 $D = 15$，静差率 $s = 5\%$，求闭环系统的开环放大系数 K；②若 $U_N^* = 10\text{V}$ 时，$n = n_N = 1500\text{r/min}$，求拖动系统测速反馈系数 α；③计算比例

调节器的放大系数 K_p；④试问系统能否稳定运行？其临界开环放大系数为多少？⑤试绘制比例调节器 K_p=20 与 K_p=21 系统的单位给定阶跃响应曲线以验证系统能否稳定运行；⑥以相角稳定裕度 $\gamma = 45°$ 为校正主要指标对系统进行滞后校正；⑦以剪切频率为校正主要指标对系统进行滞后校正。

解：（1）求满足系统调速范围与静差率要求时的闭环系统开放大系数 K

1）额定磁通下的电动机电动势转速比 $C_e = \dfrac{U_N - I_N R_a}{n_N}$。

```
>> syms UN IN nN Ra Ce;
UN=220;IN=12.5;
Ra=1; nN=1500;
Ce=(UN-IN*Ra)/nN
```

运行程序，输出结果如下：

```
Ce =
     0.1383
```

即额定磁通下的电动机电动势转速比 $C_e = 0.1383$V.min/r。

2）满足系统调整范围与静差率要求时的闭环系统稳态速降 $\Delta n_{c1} = \dfrac{n_N s}{D(1-s)}$。

```
>> syms nN s D del;
nN=1500;
s=0.05;D=15;
del=nN*s/(D*(1-s))
```

运行程序，输出结果如下：

```
del =
     5.2632
```

即满足要求时的闭环系统稳态速降 $\Delta n_{c1} = 5.2632$r/min。

3）开环系统稳态速降 $\Delta n_{op} = \dfrac{I_N R}{C_e}$。

```
>> syms IN R Ce del;
IN=12.5;
R=2.9;Ce=0.1383;
del=IN*R/Ce
```

运行程序，输出结果如下：

```
del =
     262.1114
```

即开环系统稳态速降 $\Delta n_{op} = 262.1114$r/min。

4）根据自动控制理论有 $K = \dfrac{\Delta n_{op}}{\Delta n_{c1}} - 1$。

```
>> syms delop delc1 K;
delop=262.1114;
delc1=5.2632;
K=delop/delc1-1
```



运行程序，输出结果如下：

```
K =
    48.8008
```

即满足系统调速范围与静差率要求时的闭环系统开环放大系数 K=48.8008。

（2）求系统测速反馈系数 $\alpha=\dfrac{U_N}{n_N}$

根据自动控制理论有如下方程组

$$\begin{cases}U_n=K\cdot\Delta U_n\\U_n^*-U_n=U_n\end{cases}$$

代入已知条件，得到

$$\begin{cases}U_n=48.8008\cdot\Delta U_n\\10-U_n=\Delta U_n\end{cases}$$

用 MATLAB 解此方程组代码如下：

```
>> syms Un del alp
[Un,del]=solve('Un=48.8008*del','10-Un=del');
alp=vpa(Un/1500,2)
```

运行程序，输出结果如下：

```
alp =
    6.5*10^(-3)
```

即 $\alpha=0.0065\,\text{V}\cdot\text{min/r}=K_t$ 。

根据自动控制理论，闭环系统的开环放大系数 K、测速反馈系数 α 、电动机电动势转速比 C_e 与放大系数 K_p 之间满足关系式：

$$K=\frac{K_pK_s\alpha}{C_e}\ 。$$

```
>> syms K Kp Ks Ce alp;
K=48.8008;Ks=44;
Ce=0.1383;alp=0.0065;
Kp=(K*Ce)/(Ks*alp)
```

运行程序，输出结果如下：

```
Kp =
    23.5984
```

即 K_p=23.5984。

（3）计算参数 T_a 与 T_m

1）电枢回路电磁时间常数 $T_a = L/R$。

```
>> syms L R Ta;
L=40e-3;R=2.9;
Ta=L/R
```

运行程序，输出结果如下：

```
Ta =
     0.0138
```

即电枢回路电磁时间常数 $T_a = 0.0138\text{s}$。

2）系统运动部分飞轮矩相应的电机时间常数 $T_m = \dfrac{GD^2R}{375C_eC_m}$。

```
>> syms GDpf R Ce Cm Tm;
GDpf=1.5;R=2.9;
Ce=0.1383;Cm=Ce*30/pi;
Tm=GDpf*R/(375*Ce*Cm)
```

运行程序，输出结果如下：

```
Tm =
      0.0635
```

即飞轮矩相应的电机时间常数 $T_m = 0.0635\text{s}$。

（4）绘制带参数单闭环调速系统的 Simulink 动态结构图

图 5-40 模型 xiu5_10.mdl，图中，$K_t = \alpha = 0.0065\,\text{V}\cdot\text{min/r}$。

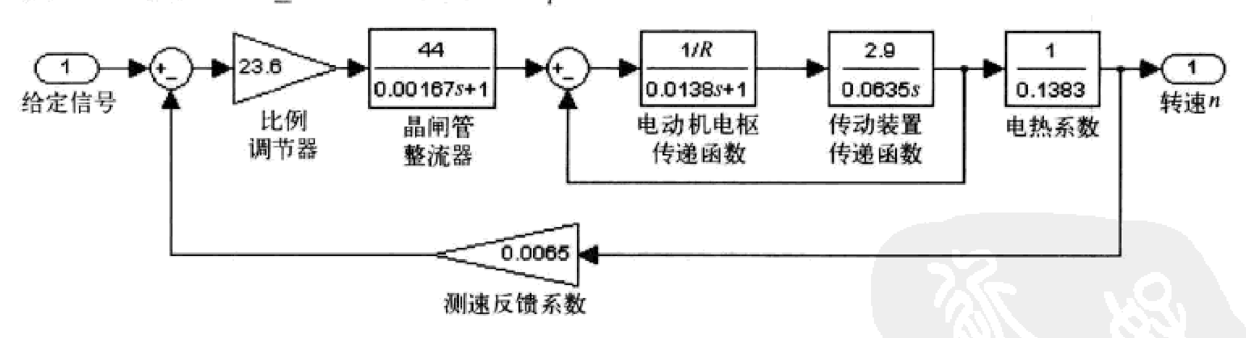


图 5-40　带参数单闭环调速系统的 Simulink 动态结构图模型

（5）求闭环系统临界开环放大系数

根据自动控制理论的代数稳定判据，系统稳定的充要条件为 $K < \dfrac{T_m(T_a+T_s)+T_s^2}{T_aT_s}$，其临界开环放大系数 $K_{cr} = \dfrac{T_m(T_a+T_s)+T_s^2}{T_aT_s}$。

```
>> syms K Kcr Tm Ta Ts;
Tm=0.0635;
Ta=0.0138;
```

```
Ts=0.00167;
Kcr=(Tm*(Ta+Ts)+Ts^2)/(Ta*Ts)
```

运行程序，输出结果如下：



```
Kcr =
      42.7464
```

即闭环系统临界开环放大系数 $K_{cr}=42.7464$。

（6）求系统闭环特征根以验证系统能否稳定运行

```
>> [a,b,c,d]=linmod('xiu5_10');
s1=ss(a,b,c,d);
sys=tf(s1);
sys1=zpk(s1);
p=sys.den{1};
roots(p)
```

运行程序，输出结果如下：

```
ans =
   1.0e+002 *
   -6.8096
   -0.0303 + 2.2577i
   -0.0303 - 2.2577i
```

即系统闭环特征根有两个根的实部为正，说明系统不能稳定运行。

（7）绘制出比例调节器 $K_p=20$ 与 $K_p=21$ 系统的单位给定阶跃响应曲线以验证系统能否稳定运行

1）比例调节器 $K_p=20$ 时，求闭环系统开环放大系数 K。

根据 $K=K_pK_s\alpha/C_e$，有：

```
>> syms Kp Ks Ce alp;
Kp=20;Ks=44;
Ce=0.1383;alp=0.0065;
K=Kp*Ks*alp/Ce
```

运行程序，输出结果如下：

```
K =
    41.3594
```

即 $K_p=20$ 对应着闭环系统开环放大系数 K=41.3594。

2）当 $K_p=20$ 时（需将动态模型结构图 xiu5_10.mdl 的 K_p 设置为 20，绘制其系统单位阶跃响应曲线。

```
%MATLAB Program xiuli5_10.m
  [a,b,c,d]=linmod('xiu5_10');
s1=ss(a,b,c,d);
```

```
sys=tf(s1);
step(sys);
```

当 $K = K_p K_s \alpha / C_e$ =41.3594< K_{cr} =42.7464，此时对应着模型 xiu5_10.mdl 中的 $K_p = 20$，程序方式下运行程序 xiuli5_10，系统单位阶跃响应曲线应呈现剧烈的振荡（虽然是衰减的），如图 5-41 所示。

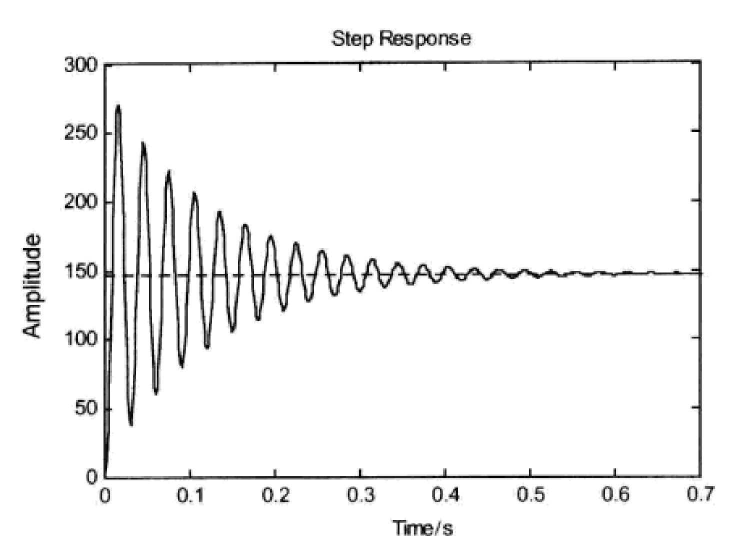


图 5-41 当 *K*p=20 时系统的单位阶跃响应

3）比例调节器 $K_p = 21$ 时，求闭环系统开环放大系数 *K*。

```
>> syms Kp Ks Ce alp;
Kp=21;kS=44;
Ce=0.1383;alp=0.0065;
K=Kp*Ks*alp/Ce
```

运行程序，输出结果如下：

```
K =
    (455*Ks)/461
```

4）当 $K_p = 21$ 时，绘制其系统单位阶跃响应曲线。

当 $K = K_p K_s \alpha / C_e$ =41.3594> K_{cr} =42.7464，此时对应着模型 xiu5_10.mdl 中的 $K_p = 21$，程序方式下运行程序 xiu5_10.m，系统单位阶跃响应呈现发散的振荡，如图 5-42 所示，即系统是不稳定的。当 $K_p = 23.5984$ 时，系统越发不稳定。

（8）分别以相角稳定裕度与剪切频率为校正主要指标对系统进行滞后校正。模型 xiu5_40.mdl 即图 5-40 的开环模型为 xiu5_40A.mdl，以下程序要用到它。

```
>> [a,b,c,d]=linmod('xiu5_10A');
s1=ss(a,b,c,d);
s2=tf(s1);
gama=48;
[Gc]=xiumg(1,s2,[gama])
wc=35;
```

```
[Gc]=xiumg(2,s2,[wc])
```

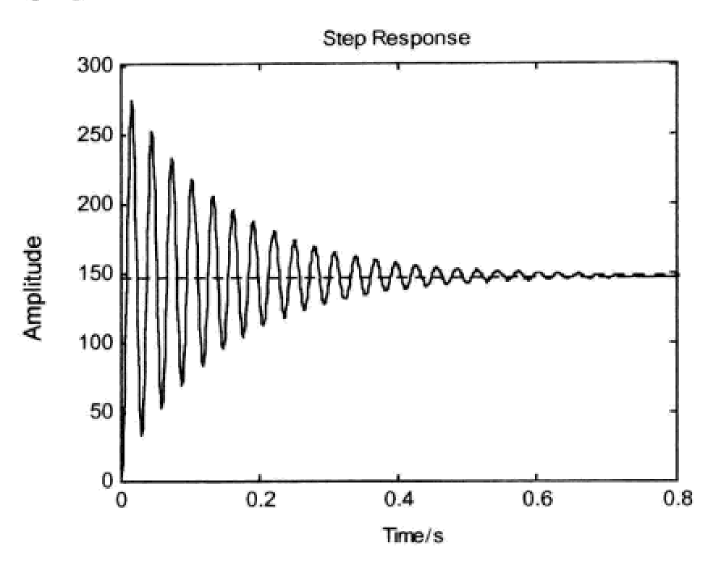


图 5-42　当 K=21 时系统的单位阶跃响应



其中，xiumg()为用户自定义函数。

```
function [Gc]=xiumg(key,sope,vars)
if key==1
      gama=vars(1);
      gama1=gama+5;
      [mu,pu,w]=bode(sope);
      wc=spline(pu,w',(gama1-180));
elseif key==2
      wc=vars(1);
end
num=sope.num{1};
den=sope.den{1};
na=polyval(num,j*wc);
da=polyval(den,j*wc);
g=na/da;
g1=abs(g);
h=20*log10(g1);
beta=10^(h/20);
T=10/wc;
betat=beta*T;
Gc=tf([T 1],[beta 1]);
```

程序运行，输出结果如下：

```
Transfer function:
0.04608 s + 1
-------------
 2098 s + 1
Transfer function:
```



```
0.2857 s + 1
------------
150.9 s + 1
```

即以相角稳定裕度为校正主要指标的滞后校正器为 $G_c(s)=\dfrac{0.1611s+1}{1.699s+1}$，而以剪切频率为校正主要指标的滞后校正器为 $G_c(s)=\dfrac{0.2857s+1}{6.26s+1}$。

以下验算设计的校正器校正效果。

1）对以相角稳定裕度为校正主要指标的校正器为 $G_c(s)=\dfrac{0.1611s+1}{1.699s+1}$。

```
>> %MATLAB Program xiuli5_10a.m
[a,b,c,d]=linmod('xiu5_10A');
s1=ss(a,b,c,d);
sys2=tf(s1);
gama=48;
[Gc]=xiumg(1,sys2,[gama]);
sys=sys2*Gc;
margin(sys)
```

程序保存为 xiuli5_10a.m，绘制出系统 Bode 图如图 5-43 所示。

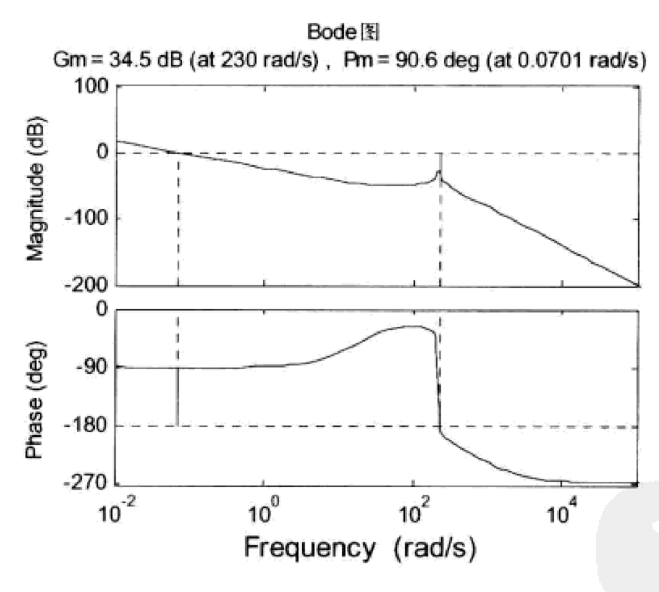


图 5-43　经校正器 $G_c(s)=\dfrac{0.1611s+1}{1.699s+1}$ 校正后的 Bode 图

从图 5-43 可以看出，校正后系统的相角稳定裕度 $\gamma=47.7^\circ\approx48^\circ$，达到预期目的。

2）对以剪切频率为校正主要指标的滞后校正器为 $G_c(s)=\dfrac{0.2857s+1}{6.26s+1}$。

```
>> %MATLAB Program xiuli5_10b.m
[a,b,c,d]=linmod('xiu5_10');
s1=ss(a,b,c,d);
sys2=tf(s1);
gama=36;
```

```
[Gc]=xiumg(2,sys2,[gama]);
sys=sys2*Gc;
margin(sys)
```

程序保存为 xiuli5_10b.m，绘制出系统 Bode 图如图 5-44 所示。



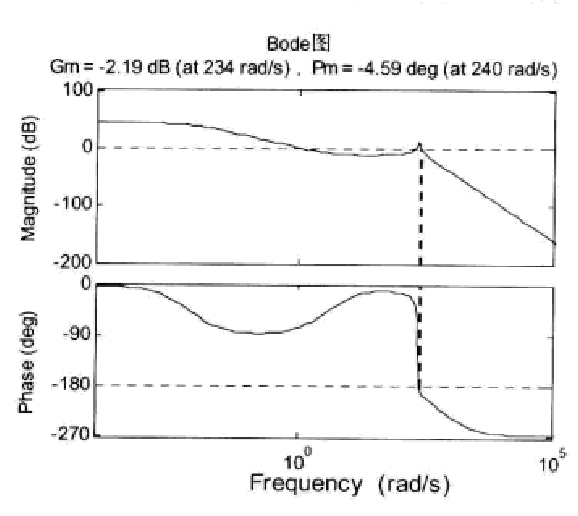


图 5-44 经校正器 $G_c(s)=\dfrac{0.2857s+1}{6.26s+1}$ 校正后的 Bode 图

5.3.2 多闭环控制系统的仿真分析

根据自动控制系统设计理论，采用两个 PI 调节器（即 ACR、ASR 均采用 PI 调节器）的双闭环调速系统具有良好的稳态与动态性能，结构简单，工作可靠，设计也很方便。实践证明，它是一种应用最广的调速系统。然而，其动态性能的不足之处就是转速超调，而且抗扰性能的提高也受到一定限制。

解决这个问题的一个简单有效的方法就是在转速调节器上引入转速微分负反馈，这样就可以抑制速度超前调直到消除超调，同时可以大大降低动态速度降落。

【例 5-11】 带转速微分负反馈的晶闸管-直流电动机双闭环调速系统（V-M 系统）的系统结构如图 5-45 所示。图 5-45 中电动机参数：P_N=3kW，nN=1500r/min，Un=220V，I_N=17.5A，电动机电枢电阻 R_a=12.5Ω，整流装置内阻 R_{rec}=1.3Ω，平波电抗器电阻 R_L=0.3Ω，V-M 系统主电路总电阻 R=2.85Ω，电枢主回路总电感 L=200mH，拖动系统运动部分飞轮矩 $GD^2=3.53\text{N}\cdot\text{m}^2$，三相桥失控平均时间 T_s=0.00167s，整流触发装置的放大系数 K_s=38，要求系统调速范围 D=20，静差率 s=10%，堵转（最大）电流 I_{dcl}=21IN，临界截止电流 I_{dcr}=2IN，ACR、ASR 均采用 PI 调节器，ASR 限幅输出 $U_{im}^*=8\text{V}$，最大给定 $U_{nm}^*=10\text{V}$。（1）试计算系统的参数：电动机电动势转速比 C_e、闭环系统稳态转速 ΔnN、触发整流装置的放大系数 K_s、电流反馈系数 β、电枢电磁时间常数 T_a、系统机电时间常数 T_m、系统测速反馈系数 α；（2）选择几个滤波时间长度 T0i、T0n、T0dn 与中频宽 h；（3）计算电流调节器传递函数 $W_{ACR}(s)=K_i\dfrac{\tau_i s+1}{\tau_i s}$；（4）计算转速调节器传递函数 $W_{ASR}(s)=K_n\dfrac{\tau_n s+1}{\tau_n s}$；（5）对双闭环调速系统进行单位阶跃给定响应仿真与单位阶跃负载扰动响应仿真。（6）对转速微分负反馈环节 $\dfrac{\alpha\tau_{dn}s}{T_{0dn}s+1}$ 进行参数计算；（7）对带转速微分负反馈双闭环调速系统进行单位阶跃

响应仿真与单位阶跃负载扰动响应仿真，对（5）与（7）两项仿真作简单比较；（8）计算退饱和时间t_t退饱和转速n_t。

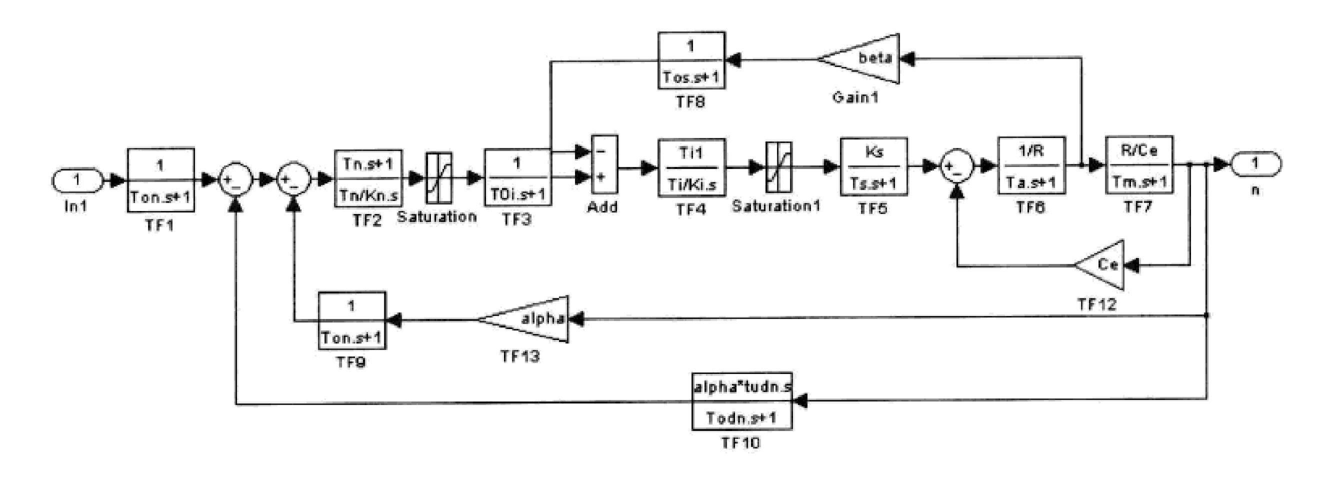


图 5-45　带转速微分负反馈的 V-M 双闭环系统结构图模型 xiu5_11.mdl

解：（1）拖动调速系统几个参数的计算与选择

1）额定磁通下的电动机电动势转速比$C_e = \dfrac{U_N - I_N R_a}{n_N}$。

```
>> syms UN IN nN Ra Ce;
UN=220;IN=17.5;
Ra=1.251;nN=1500;
Ce=(UN-IN*Ra)/nN
```

运行程序，输出结果如下：

```
Ce =
        0.1321
```

即额定磁通下的电动机电动势转速比$C_e = 0.1324\text{V}\cdot\text{min/r}$。

2）满足系统调速范围与静差率要求的闭环系统稳态速降$\Delta n_N = \dfrac{n_N}{D(1-s)}$。

```
>> syms nN s d del;
nN=1500;s=0.1;
d=20;del=nN*s/(d*(1-s))
```

运行程序，输出结果如下：

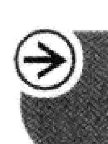
```
del =
       8.3333
```

即满足要求的闭环磁通稳态速降$\Delta n_N = 8.3333\text{r/min}$。

3）满足系统要求的触发整流装置的放大系数$K_s = \dfrac{C_e n_N + I_{dbl} R}{U_{ctm}}$。

```
>> syms Ks nN Ibdl R Ce Uctm
```

```
Ce=0.1321; nN=1500;
Ibdl=2.1*17.5; R=2.85;
 Uctm=8;
 Ks=(Ce*nN+Ibdl*R)/Uctm
```



运行程序，输出结果如下：

```
Ks =
    37.8609
```

即取系统要求的触发整流装置的放大系数 $K_s = 38$。

4）满足系统要求的电流反馈系数 $\beta = \frac{U_{im}^*}{I_{dm}} = \frac{U_{im}^*}{2.1I_N}$。

```
>> syms beta Uim Idm IN;
 Uim=8;IN=17.5;
 Idm=21*IN;
 beta=Uim/Idm
```

运行程序，输出结果如下：

```
beta =
      0.0218
```

即满足系统要求的电流反馈系统 $\beta = 0.2177$V/A。

5）电动机电枢电磁时间常数 $T_a = \frac{L}{R}$。

```
>>   syms Ta R L;
 L=200*10^(-3);
 R=2.85;
 Ta=L/R
```

运行程序，输出结果如下：

```
Ta =
    0.0702
```

即电动机电枢电磁时间常数 $T_a = 0.0702$s。

6）电机拖动系统电机时间常数 $T_m = \frac{GD^2R}{375C_eC_m}$。

```
>>   syms GDpf R Ce Cm;
 GDpf=3.53; R=2.85;
 Ce=0.1321; Cm=30*Ce/pi;
 Tm=(GDpf*R)/(375*Ce*Cm)
```

运行程序，输出结果如下：

```
Tm =
```

```
0.1610
```

7）满足系统要求的转速反馈系数 $\alpha=\dfrac{U_n^*}{n_N}$。

```
>>  syms alp Un nN;
 Un=10;nN=1500;
 alp=Un/nN
```

运行程序，输出结果如下：

```
alp =
     0.0067
```

即满足系统要求的转速反馈系统 $\alpha=0.0067\text{V}\cdot\text{min/r}$。

（2）选取电流环滤波时间常数 $T_{0i}=0.002\text{s}$；选取转速环滤波时间常数 $T_{0n}=0.01\text{s}$；选取转速微分滤波时间常数 $T_{0dn}=T_{0N}=0.01\text{s}$；选择中频宽 $h=5$。

（3）电流调节器 $W_{ACR}(s)=K_i\dfrac{\tau_i s+1}{\tau_i}$ 参数计算

根据自动控制系统设计理论，选取积分时间常数 $\tau_i=T_a=0.0702\text{s}$；三相桥整流电路平均失控时间 $T_s=0.00167\text{s}$；合并电流环小时间常数 $T_{\Sigma i}=T_s+T_{0i}=0.00167+0.002=0.00367\text{s}$；电流环开环增益 $K_I=\dfrac{1}{2T_{\Sigma i}}=\dfrac{1}{2\times0.00367}=136.2398s^{-1}$；电流调节器的比例系数 $K_i=K_I\dfrac{\tau_i R}{\beta K_s}$ =3.2904，所以电流调节器的传递函数为：

$$W_{AcR}(s)=K_i\frac{\tau_i s+1}{\tau_i}=3.2904\times\frac{0.0702s+1}{0.0702s}=\frac{0.0702s+1}{0.0213s}$$

（4）转速调节器 $W_{AsR}(s)=K_n\dfrac{\tau_n s+1}{\tau_n}$ 参数的计算

根据自动控制系统设计理论，$T_{\Sigma i}=0.00367\text{s}$，合并转速环小时间常数 $T_{\Sigma n}=2T_{\Sigma i}+T_{0n}=2\times0.00367+0.01=0.0173\text{s}$；选取转速调节器积分时间常数 $\tau_n=hT_{\Sigma n}=5\times0.00367+0.01=0.0173s$；转速开环增益 $K_N=\dfrac{h+1}{2h^2T_{\Sigma n}^2}=\dfrac{6}{50\times0.0173s^2}=400.95s^{-2}$；转速调节器比例系数 $K_n=\dfrac{(h+1)\beta C_e T_m}{2h\alpha RT_{\Sigma n}}=\dfrac{6\times0.2177\times0.1321\times0.1601}{10\times2.85\times0.0067\times0.0173}=8.4095s^{-2}$，所以转速调节器传递函数 $W_{AsR}(s)=K_n\dfrac{\tau_n s+1}{\tau_n}=8.4095\times\dfrac{0.0867s+1}{0.0867s}=\dfrac{0.0867s+1}{0.0103s}$。

（5）双闭环调速系统的 Simulink 动态结构图及其仿真

采用两个 PI 调节器的双闭环调速系统原理如图 5-46 所示。

带参数的双闭环调速系统原理图如图 5-47 所示，并另存模型为 xiu5_11A.mdl。

用以下 MATLAB 程序绘制双闭环调速系统的单位阶跃响应曲线。

```
>> %MATLAB Program xiuli5_11A.m
[a,b,c,d]=linmod('xiu5_11A');
```

```
s1=ss(a,b,c,d);
sys=tf(s1);
step(sys)
```



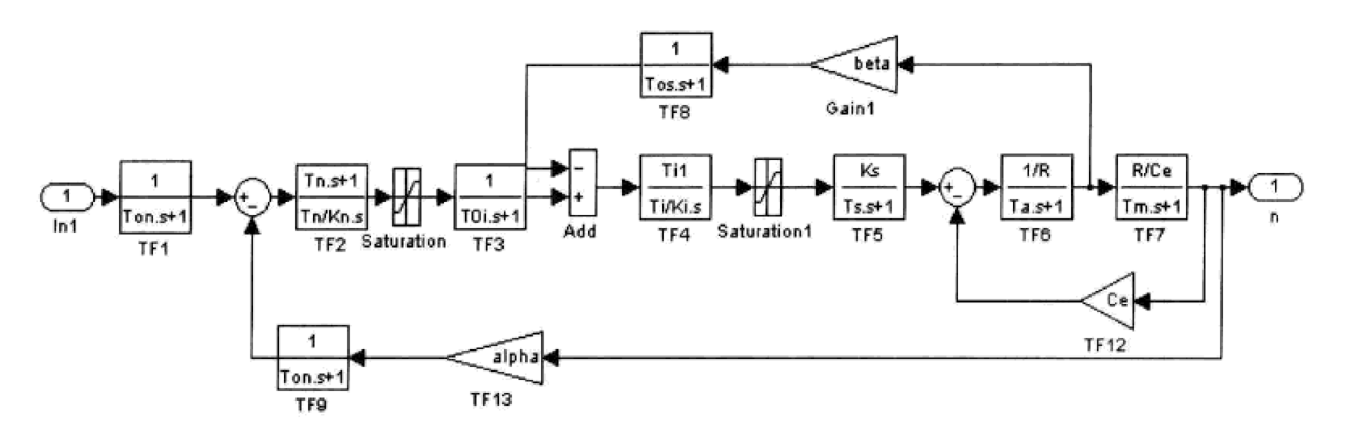


图 5-46　双闭环调速系统原理图

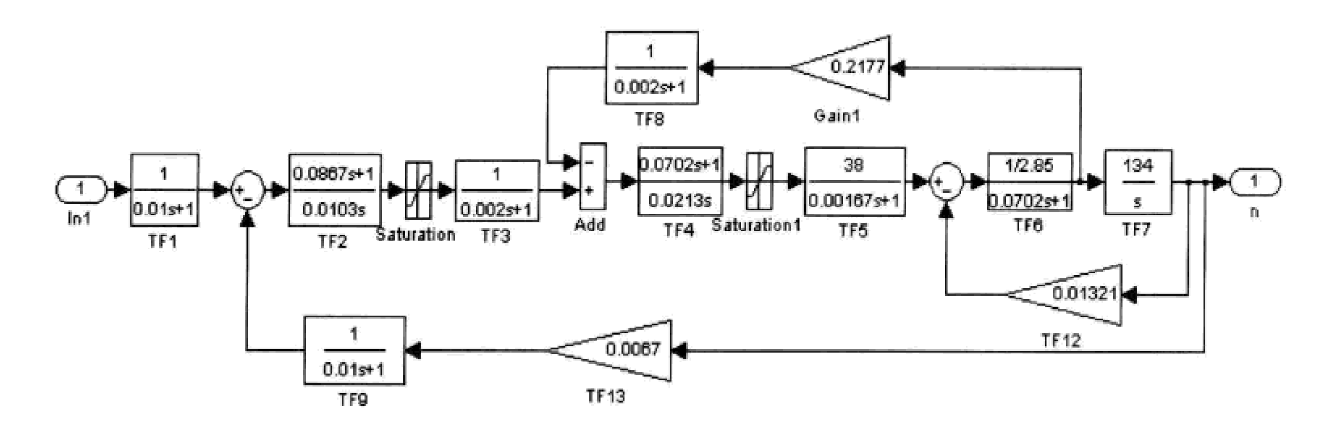


图 5-47　带参数的双闭环调速系统结构图模型 xiu5_1A.mdl

程序运行后，绘制的单位阶跃响应曲线如图 5-48 所示。

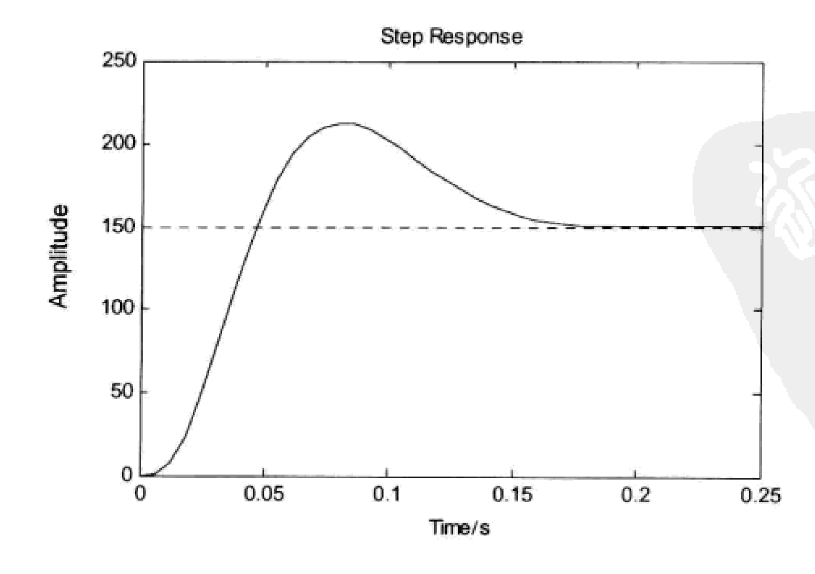


图 5-48　双闭环调速的单位阶跃响应曲线

双闭环调速系统负载扰动仿真动态结构如图 5-49 所示，并另存模型为 xiu5_11B.mdl。

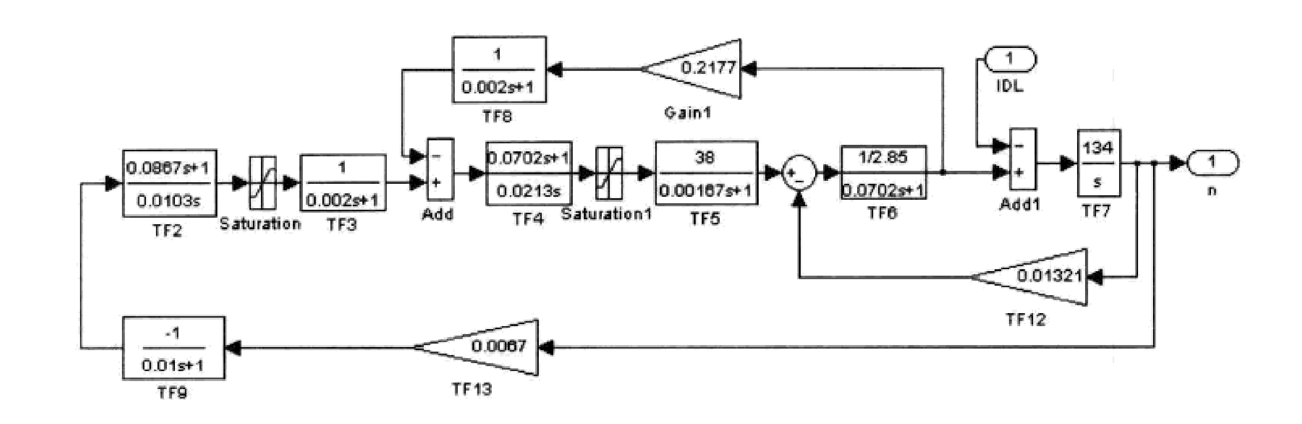


图 5-49　双环调速系统负载扰动仿真动态结构图 xiu5_11B.mdl

用以下 MATLAB 程序绘制双环调速系统的单位阶跃负载扰动响应仿真曲线。

```
>> %MATLAB Program xiuli5_11B.m
[a,b,c,d]=linmod('xiu5_11B');
s1=ss(a,b,c,d);
t1=[0:0.001:0.3];
[y,t]=step(s1,t1);
step(s1,t1);
[de,tp,tv]=dist(y,t)
```

运行程序，绘制的单位阶跃负载扰动响应曲线如图 5-50 所示，并计算出性能指标。

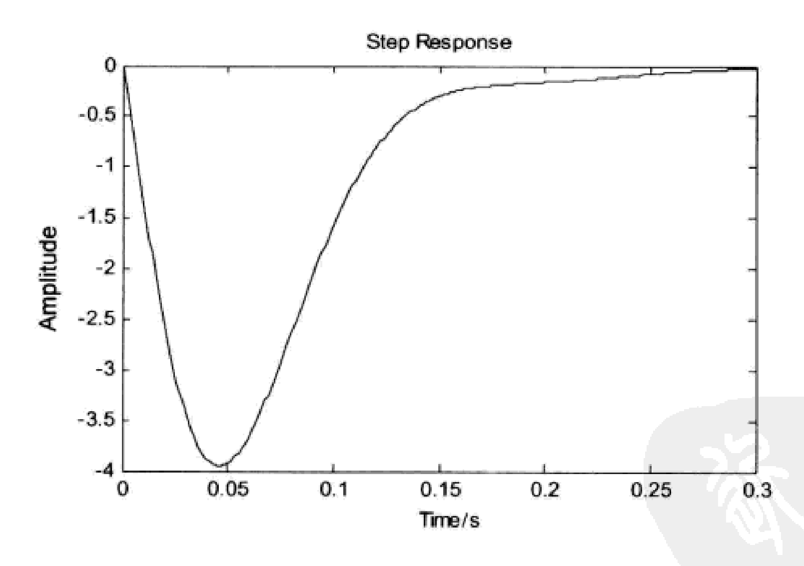


图 5-50　双闭环调速系统的单位阶跃负载扰动响应曲线

最大动态降落 $\Delta c_{max}\% = -3.934\%$；最大动态降落时间 $t_p = 0.0460\text{s}$；恢复时间 $t_v =$ 0.2720s（对应 5%的误差带）。

de=-3.9347；tp=0.0460；tv=0.2720

（6）转速微分负反馈环节 $\dfrac{\alpha\tau_{dn}s}{T_{0dn}s+1}$ 参数的计算

已知计算出转速反馈系数 $\alpha = 0.0067\text{V}\cdot\text{min/r}$。根据自动控制系统设计原理论，选取转速微分滤波时间常数 $T_{0dn} = T_{0n} = 0.01\text{s}$。

$$\tau_{dn}\big|_{\sigma=0} \geqslant \frac{4h+2}{h+1}T_{\Sigma n} = \frac{20+2}{6}\times 0.0173 = 0.0634s$$

取 $\tau_{dn} = 0.0634s$ 那么，$\dfrac{\alpha\tau_{dn}s}{T_{0dn}s+1} = \dfrac{0.0067\times 0.0634s}{0.01s+1}$。



（7）带参数转速微分负反馈双闭环调速系统的 Simulink 动态结构图

根据图 5-45 可知，将已知参数与算得的参数代入图 5-45 中即得带参数的 Simulink 动态结构图，如图 5-51 所示，即系统动态模型 xiu5_11C.mdl，以下仿真要用到它。

用以下 MATLAB 程序绘制转速微分负反馈双闭环调速系统的单位阶跃响应曲线。

```
>> % MATLAB Program xiuli5_11C.m
clear all;
[a b c d]=linmod('xiu5_11C');
s1=ss(a,b,c,d);
sys=tf(s1);
step(sys)
```

运行程序，绘制的单位阶跃响应曲线如图 5-52 所示。

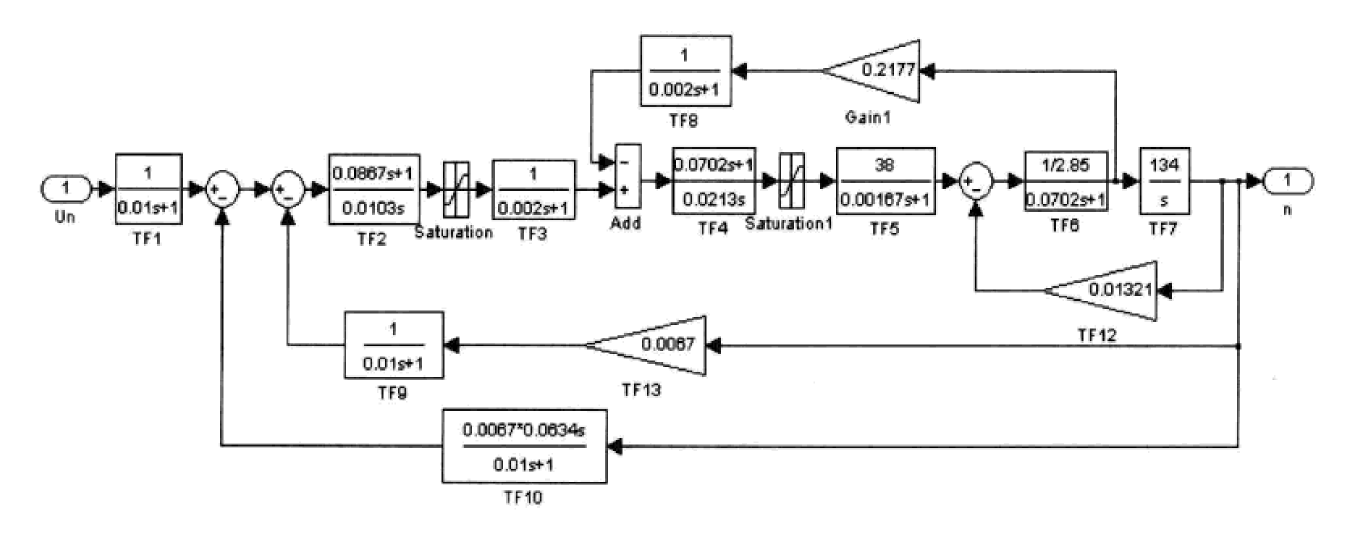


图 5-51 带参数传递微分负反馈的 V-M 双闭环系统结构图模型 xiu5_11C.mdl

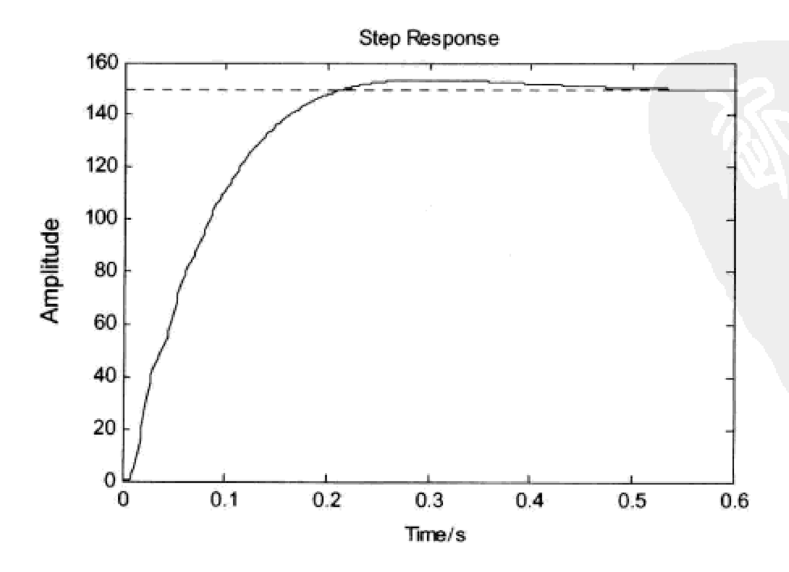


图 5-52 转速微分负反馈双闭环调速系统的单位阶跃响应曲线

比较图 5-50 与图 5-52 可见，普通双闭环系统单位阶跃响应的超调$\sigma\% > 30\%$；而带转速微分负反馈的双闭环系统，其单位阶跃响应基本无超调，转速微分负反馈作用显著。

转速微分负反馈双闭环调速系统负载扰动仿真动态结构图如图 5-53 所示，即系统动态模型 xiu5_11D.mdl。

用以下 MATLAB 程序绘制双闭环调速系统的单位阶跃负载扰动的响应仿真曲线。

```
>> % MATLAB Program xiuli5_11D.m
[a,b,c,d]=linmod('xiu5_11D');
s1=ss(a,b,c,d);
t1=[0:0.001:0.3];
[y,t]=step(s1,t1);
step(s1,t1);
[de,tp,tv]=dist(1,y,t)
```

运行程序，绘制的单位阶跃负载扰动响应曲线如图 5-54 所示，并计算出性能指标。

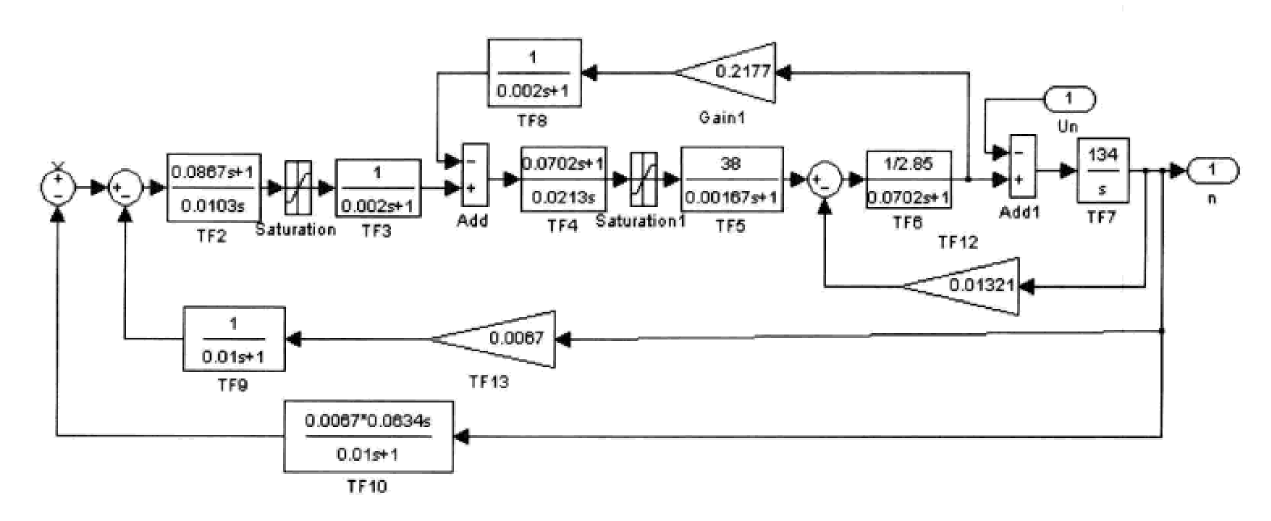


图 5-53 转速微分负反馈双闭环调速系统负载扰动仿真动态模型 xiu5_11D.mdl

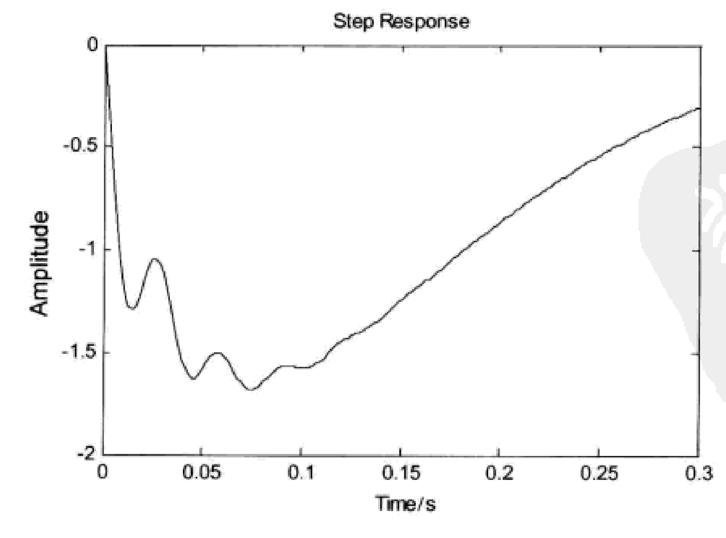


图 5-54 双闭环调速系统的单位阶跃负载扰动响应曲线

```
de=-1.6788
tp=0.0750
```

```
tv=0.3000
```

即最大动态 $c_{\max}\% = -1.6788\%$；最大动态降落时间 $t_p = 0.0750\text{s}$；恢复时间 $t_v = 0.30\text{s}$（对应 5%的误差带）。



比较图 5-50 与图 5-54 及其计算的数据，对于单位阶跃负载扰动，带转速微分负反馈的双闭环系统比普通双闭环系统，其最大动态降落要小得多，但最大动态降落时间与恢复时间要长一些。

（8）计算退饱和时间 t_t 与退饱和转速 n_t

根据自动控制系统设计理论，有退饱和时间计算公式，即

$$t_t = \frac{C_e n^* T_m}{R(I_{dm} - I_{dL})} + T_{\Sigma n} - \tau_{dn}$$

还有退饱和转速计算公式 $n_t = n^* - \dfrac{R}{C_e T_m}(I_{dm} - I_{dL})\tau_{dn}$。这可以用以下 MATLAB 程序计算退饱和时间 t_t 与退饱和转速 n_t。

```
>> syms Ce nx Tm R Idm IN IdL Tsigman taudn tt;
Ce=0.1321;nx=1500;
Tm=0.1610;R=2.85;
IN=17.5;Idbl=2.1*IN;
Idm=Idbl;IdL=0;
Tsigman=0.0173;taudn=0.0634;
tt=Ce*nx*Tm/(R*(Idm-IdL))+Tsigman-taudn
nt=nx-R*(Idm-IdL)*taudn/(Ce*Tm)
```

运行程序，输出结果如下：

```
tt =
     0.2585
nt =
   1.1878e+003
```

5.4 PID 控制器的微积分分析

PID（比例、积分、微分）控制是发展较早、理论成熟、应用广泛的一种控制策略，在自动控制原理、过程控制等课程中都有讲述。PID 控制器调节方便、控制效果较好，在工农业生产中有着广泛的应用基础。

1. PID 控制器的连续表达式

PID 控制器由比例环节（Proportional）、积分环节（Integral）和微分环节（Differential）组成，连续 PID 控制器的一般形式为：

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau)\mathrm{d}\tau + K_D \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{5-15}$$

或

$$u(t)=K_p\left[e(t)+\frac{1}{T_I}\int_0^t e(\tau)\mathrm{d}\tau+T_D\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right] \quad (5\text{-}16)$$

连续 PID 控制器的传递函数模型为：

$$Gc(s)=K_p+\frac{K_I}{s}+K_D s \quad (5\text{-}17)$$

或

$$Gc(s)=K_p\left(1+\frac{1}{T_I s}+\frac{T_D s}{\frac{T_D}{N}+1}\right) \quad (5\text{-}18)$$

式中，N 趋于无穷大时，则为纯微分运算。实际的仿真设计中，N 不必过大，取 N=10，就可以逼近实际的微分效果。

2. PID 控制器的离散表达式

如果采用时间 T 足够小，在第 k 个采样周期的误差 $e(t)$ 的导数与积分可以表示为：

$$\frac{\mathrm{d}e(t)}{\mathrm{d}t}=\frac{e(kT)-e\left[(k-1)T\right]}{T} \quad (5\text{-}19)$$

和

$$\int_0^{kT}e(t)\mathrm{d}t\approx T\sum_{i=0}^{k}e(iT)=\int_0^{(k-1)T}e(t)\mathrm{d}t+Te(kT) \quad (5\text{-}20)$$

由式（5-15）、式（5-19）和式（5-20）可知，离散形式的 PID 表达式为：

$$u(kT)=K_p e(kT)+K_I\sum_{i=0}^{k}e(iT)+K_D\{e(kT)-e[(k-1)T]\} \quad (5\text{-}21)$$

所以离散 PID 控制器的脉冲传递函数为：

$$Gc(z)=K_p+\frac{K_I}{1-z^{-1}}+K_D(1-z^{-1}) \quad (5\text{-}22)$$

5.4.1 比例控制及性能分析

比例（P）控制是一种最简单的控制方式，当式（5-15）中的积分系数和微分系数为零时，PID 控制器的一般形式便简化为 P 控制器。控制器的输出与输入误差成比例关系，当输入误差为零时，控制器输出为零。纯比例控制器属于有差调节。比例控制器的传递函数为：

$$G_p(s)=K_p \quad (5\text{-}23)$$

对于单位反馈系统，当输入为阶跃信号 $R_0(t)$ 时，0 型系统响应的稳态误差与其开环增益 K 近似成反比关系，即

$$\lim_{t\to\infty}e(t)=\frac{R_0}{1+K} \quad (5\text{-}24)$$

对于单位反馈系统，当输入为斜坡信号 $R_I t$ 时，I 型系统响应的稳态误差与其开环增益 K_V 近似成反比关系，即

$$\lim_{t\to\infty} e(t) = \frac{R_1}{K_V} \quad (5\text{-}25)$$

比例控制只改变系统的增益而不影响相位，它对系统的影响主要反映在系统的稳态误差、上升时间和稳定性上。增大比例系数可提高系统的开环增益，减小系统的稳态误差，加快系统的响应速度，但增大比例系数会使最大超调量增大，同时降低系统的稳定性，严重时可造成闭环系统不稳定。此外，纯比例控制器属于有差调节，单独作用不能消除稳态误差，所以一般不单独使用。





【例 5-12】 已知某单位反馈系统开环传递函数如下：

$$G_K(s) = \frac{1}{(s+1)(2s+1)}$$

如果采用比例（P）控制器进行调节，试绘制比例系数 K_p 分别为 1、4、10、50 时的单位阶跃响应曲线，并分析比例控制器对控制系统性能的影响。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
num=1;
den=conv([1 1],[2 1]);
Gk=tf(num,den);                    %生成开环传递函数
Kp=1;
sys=feedback(Kp*Gk,1,-1);          %生成 Kp=1 的单位负反馈闭传递函数
step(sys,'r:');                    %绘制 P 控制作用下闭环系统单位阶跃响应曲线
hold on
gtext('Kp=1');
pause
Kp=4;
sys=feedback(Kp*Gk,1,-1);          %生成 Kp=4 的单位负反馈闭环传递函数
step(sys,'b-');                    %绘制 P 控制作用下闭环系统单位阶跃响应曲线
hold on;
gtext('Kp=4');
pause
Kp=10;
sys=feedback(Kp*Gk,1,-1);          %生成 Kp=10 的单位负反馈闭环传递函数
step(sys,'k--');                   %绘制 P 控制作用下闭环系统单位阶跃响应曲线
hold on;
gtext('Kp=10');
pause
Kp=50;
sys=feedback(Kp*Gk,1,-1);          %生成 Kp=50 的单位负反馈闭环传递函数
step(sys,'m-');                    %绘制 P 控制作用下闭环系统单位阶跃响应曲线
hold on;
gtext('Kp=50');
title('比例控制性能分析');
xlabel('时间(秒)');ylabel('幅值')
```

运行以下程序，可得到不同比例系数下闭环系统单位阶跃响应曲线，如图 5-55 所示。

从图中可以看出，随着比例系数的增加，闭环系统稳态误差减小，上升时间缩短，调节次数增大，最大超调量增大，而且闭环系统稳态误差无法消除。

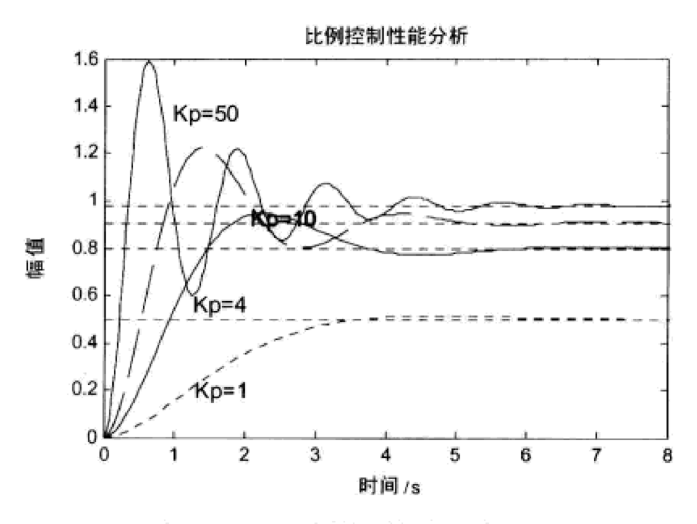


图 5-55 比例控制性能分析图

5.4.2 比例积分控制及性能分析

积分控制一般不单独使用，通常以比例积分（PI）控制器的形式出现，为了研究积分环节对系统性能的影响，可固定比例系数为常值。PI 控制器的传递函数为：

$$G_c(s) = K_p + \frac{K_I}{s} \tag{5-26}$$

或

$$G_c(s) = K_p\left(1 + \frac{1}{T_I s}\right) \tag{5-27}$$

根据 PID 控制理论，积分系数 K_I 越大（或积分时间常数 T_I 越小），积分速度越快，系统响应速度越快，调节时间缩短，同时系统的最大超调量增大；积分系数 K_I 越小（或积分时间常数 T_I 越大），积分速度越慢，系统响应速度越慢，上升时间及调节时间延长，同时系统的最大超调量减小。积分控制在稳态时误差为零。

【例 5-13】 已知某单位反馈系统开环传递函数如下：

$$G_K(s) = \frac{1}{(s+1)(s+2)}$$

如果采用比例积分（PI）控制器进行调节，试绘制比例系数 K_p=1，积分系数 K_I 为 0.2、0.8、1.4、20、5 时的单位阶跃响应曲线，并分析积分控制环节对控制系统性能的影响。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
num=1;
den=conv([1 1],[1 2]);
Gk=tf(num,den);                         %生成开环传递函数
Kp=1;
for Ki=0.2:1:2.2;
    Gc=tf([Kp,Ki],[1 0]);               %生成 PI 控制器传递函数 Gc(s)=[Kps+Ki]/s
```

```
        sys=feedback(Gc*Gk,1,-1);            %生成单位负反馈传递函数
        step(sys);
        hold on;
    end
    Gc=tf([Kp,5],[1 0]);                     %生成 Ki=5 的 PI 控制传递函数 Gc(s)=[Kps+Ki]/s
    sys=feedback(Gc*Gk,1,-1);                %生成 Ki=5 的单位负反馈闭环传递函数
    step(sys);                               %绘制 Ki=5 的闭环系统单位阶跃响应曲线
    title('积分控制性能分析');
    xlabel('时间(秒)');
    ylabel('幅值');
    axis([0 60 0 1.6]);
    gtext('Ki=0.2');gtext('Ki=1.2');
    gtext('Ki=2.2');gtext('Ki=5');
```

执行以下程序，可得到不同积分系数下闭环系统单位阶跃响应曲线，如图 5-56 所示。从图中可以看出，随着积分系数的增大，闭环系统响应速度加快，调节次数增加，最大超调量增大，稳定性变差。同时由于积分环节的存在，闭环系统稳态误差为零。

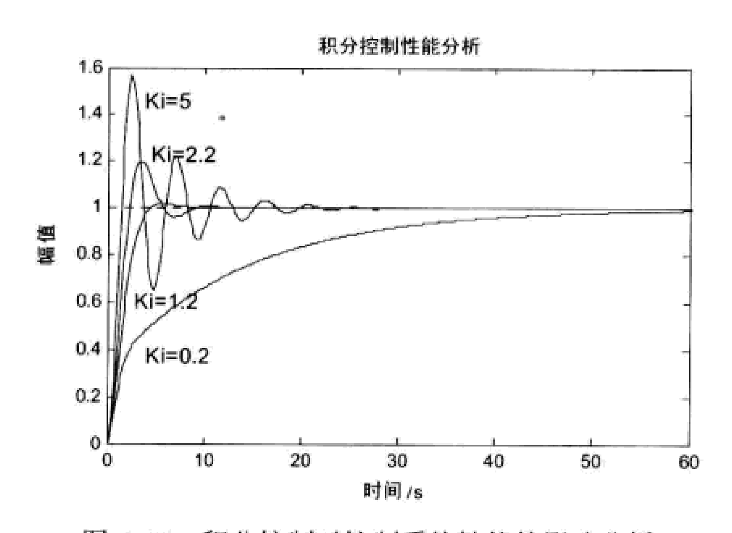


图 5-56　积分控制对控制系统性能的影响分析



5.4.3 比例微分控制及性能分析

微分（PD）控制通常与比例控制一起使用，构成比例微分控制。为了研究微分控制对系统性能的影响，可设定比例微分控制器的比例系数为常值。PD 控制器传递函数为：

$$G_c(s)=K_p+K_Ds \tag{5-28}$$

或

$$G_c(s)=K_p(1+K_Ds) \tag{5-29}$$

根据 PID 控制理论，微分控制的输出与系统偏差的变化率成正比。当微分系数增大时，闭环系统响应速度加快，系统的稳态性能变差，严重时系统不能稳定。微分环节主要作用是加快系统的响应速度。当系统偏差无变化时，微分控制器的输出为零。

【例 5-14】 已知某单位反馈系统开环传递函数如下：

$$G_K(s)=\frac{1}{(s+1)(2s+1)}$$

如果采用比例微分控制器进行调节，试绘制比例系数为 $K_p=1$，微分系数 K_p 分别为 0.3、1.8、3.3、9.8 时的单位阶跃响应曲线，并分析微分控制对控制系统性能的影响。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
num=1;
den=conv([1 1],[1 2]);
Gk=tf(num,den);                     %生成开环传递函数
Kp=1;
for Kd=0.3:1.5:3.3;
    Gc=tf([Kd*Kp,Ki],1);            %生成 PD 控制器传递函数 Gc(s)=[Kp*Kd*s+Ki]
    sys=feedback(Gc*Gk,1,-1);       %生成单位负反馈传递函数
    step(sys);
    hold on;
end
axis([0 10 0 1]);
gtext('Kd=0.3');gtext('Kd=1.8');
gtext('Kd=3.3');pause(1);
Kd=9.8;
Gc=tf([Kp*Kd,Kp],1);                %生成 Kd=9.8 的 PD 控制器传递函数 Gc(s)=[Kp*Kd*s+Kp]
sys=feedback(Gc*Gk,1,-1);           %生成 Kd=9.8 的单位负反馈闭环传递函数
step(sys);                          %绘制 Kd=9.8 的闭环系统单位阶跃响应曲线
title('积分控制性能分析');
xlabel('时间(秒)');
ylabel('幅值');
axis([0 60 0 1.6]);
gtext('K=9.8');
```

运行程序，可得不同微分系统下闭环系统单位阶跃响应曲线，如图 5-57 所示。从图中可以看出，随着微分系数的增大，闭环系统上升时间减小，最大超调量减小，调节时间减小，同时比例微分控制无法消除稳态误差。

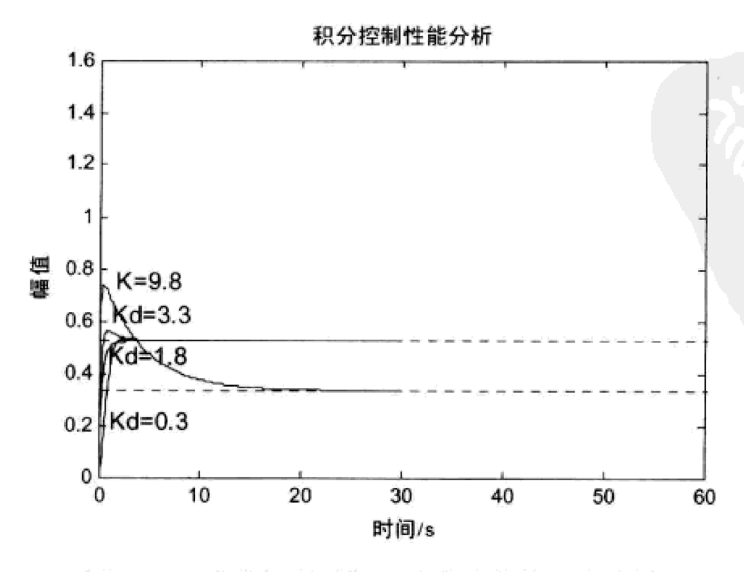
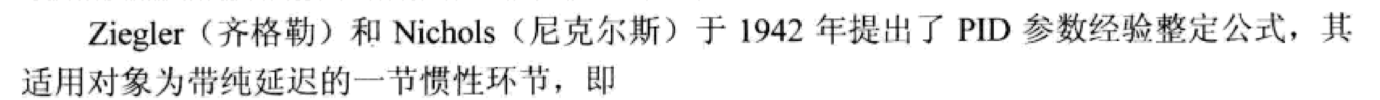


图 5-57　微分控制对闭环系统性能的影响分析

5.5 Simulink 仿真在 PID 控制器中的设计



5.5.1 Ziegler-Nichols 整定法及其 MATLAB 实现

Ziegler（齐格勒）和 Nichols（尼克尔斯）于 1942 年提出了 PID 参数经验整定公式，其适用对象为带纯延迟的一节惯性环节，即

$$G(s)=\frac{K}{Ts+1}\mathrm{e}^{-\tau s} \tag{5-30}$$

式中，K 为比例系数；T 为惯性时间常数；τ 为纯延迟时间常数。

在实际的工业过程中，大多数被控对象数学模型可近似为式（5-30）所示的带纯延迟的一阶惯性环节。在获得被控对象的近似数学模型后，可通过时域或频域数据，根据表 5-1 所示的 Ziegler-Nichols 经验整定公式计算 PID 参数。

表 5-1 PID 参数的 Ziegler-Nichols 经验整定公式

控制器类型	由阶跃响应整定			由频域响应整定		
	Kp	T_i	T_d	Kp	T_i	T_d
P	$T/K\tau$	无	无	0.5 Kc	无	无
PI	$0.9T/K\tau$	3 τ	无	0.4 Kc	0.8 Tc	无
PID		2 τ	0.5 τ	0.6 Kc	0.5 Tc	0.12 Tc

Ziegler-Nichols 整定的时域分析方法根据给定对象时域响应来确定 PID 控制器的参数。

如果单位阶跃响应曲线看起来近似一条 S 形曲线，则可用 Ziegler-Nichols 经验整定公式，否则，该公式不适用。由 S 形曲线可获取被控对象数据模型（式 5-30）的比例系数 K、时间常数 T 和纯延迟时间 τ。

如果被控对象不含有纯延迟环节，就不能够通过 Ziegler-Nichols 时域整定公式进行 PID 参数的整定，此时可求取被控对象的频域响应数据，通过表 5-1 所示的 Ziegler-Nichols 频域整定公式设计 PID 参数。如果被控对象含有纯延迟环节，可通过 Pade 命令将纯延迟环节近似为一个四阶传递函数模型，然后求取被控对象的频域响应数据，应用表 5-1 求取 PID 控制器的参数。在表 5-1 中，K_c 为被控对象幅值裕量，ω_c 为截止频率（或剪切频率），$T_c=2*pi/\omega_c$。

【例 5-15】 已知被控对象的数学模型如下：

$$G(s)=\frac{25}{s^2+7s+25}$$

试用 Ziegler-Nichols 整定时域整定方法设计一个 PID 控制器，并设计在 PID 控制器作用下系统的单位阶跃响应曲线。

首先求取被控对象的单位阶跃响应曲线，获取近似 4 的开环放大系数、时间常数和延迟时间。其实现的 MATLAB 程序代码如下：

```
>> num=25;
den=[1 7 25];
```

```
Gk=tf(num,den);
step(Gk)
title('开环阶跃响应曲线');
xlabel('时间(秒)');ylabel('响应');
grid on;
```

运行程序，效果如图 5-58 所示。

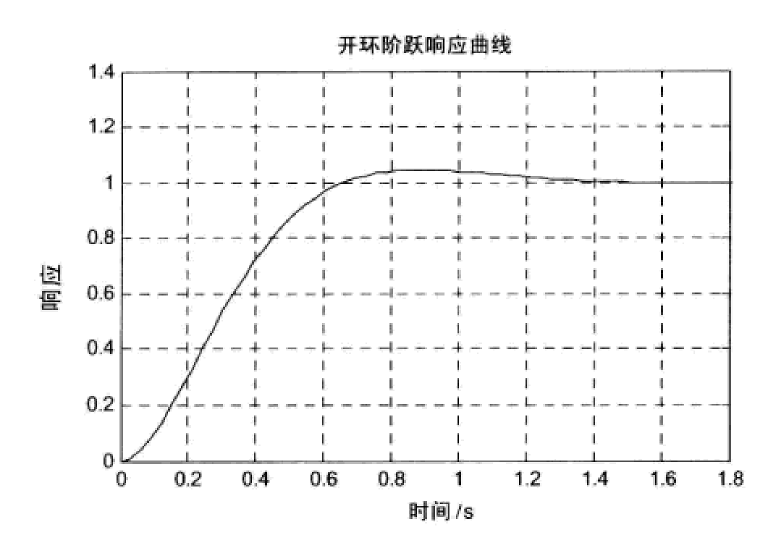


图 5-58　*S* 形响应曲线

由图 5-58 可以看出，被控对象近似 *S* 形曲线，纯延迟时间 $\tau = 0.05$，增益 K=1，时间常数 0.5。由此可得被控对象近似带纯滞后的一节惯性环节传递函数模型为：

$$G(s) = \frac{1}{0.45s + 1}\mathrm{e}^{-0.05s}$$

由表 5-1 的 PID 参数的 Ziegler-Nichols 经验整定公式，可得 PID 控制器的参数。并将 PID 控制器加在真实对象数学模型上，可得其阶跃响应曲线。实现系统闭环阶跃响应曲线的 MATLAB 代码如下：

```
>> K=1;
T=0.45;
tao=0.05;
num=25;
den=[1 7 25];
G=tf(num,den);
s=tf('s');
%PID 控制器设计
Kp=1.2*T/(K*tao);
Ti=2*tao;
Td=0.5*tao;
Gc=Kp*(1+1/(Ti*s)+Td*s);
Gk=Gc*G;
sys=feedback(Gk,1,-1);
step(sys,'r--')   %求 PID 控制作用下系统单位阶跃响应,线形为红色虚线
```

```
title('PID 控制单位阶跃响应');
xlabel('时间(秒)');ylabel('幅值');
grid on;
```

运行程序，输出效果如图 5-59 所示。



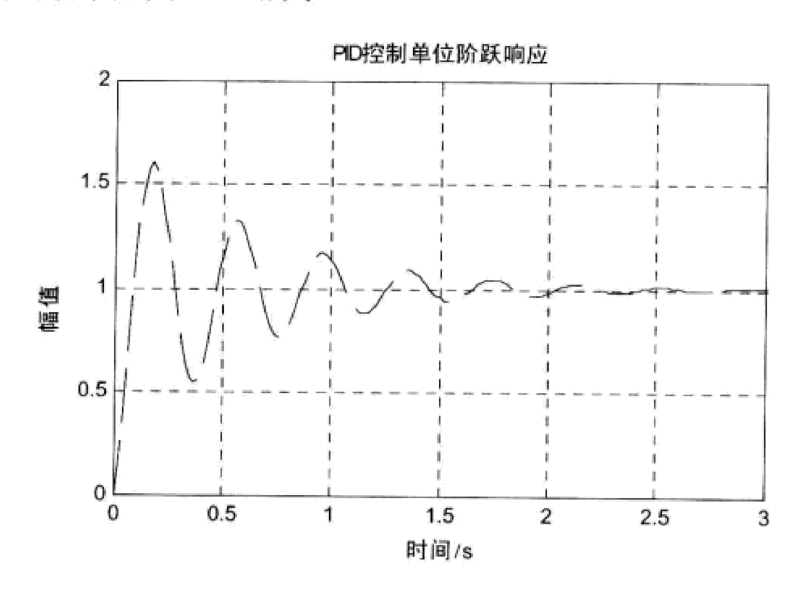


图 5-59 Ziegler-Nichols 整定法的单位阶跃响应曲线

5.5.2 Ziegler-Nichols 整定法的 Simulink 仿真设计

在前面已经介绍了 PID 控制器的 Ziegler-Nichols 整定方法，由此可以得出 PID 控制器参数的初始值。由于不同系统对控制性能的要求不同，因此由 Ziegler-Nichols 整定法得到的 PID 参数还需要进一步整定。利用 MATLAB 命令行仿真技术虽然能够实现这一点，但是不够灵活。Simulink 智能化的仿真环境，有助于 PID 控制器参数整定的后期调试。下面以示例说明。

【例 5-16】 已知控制系统框图如图 5-60 所示。

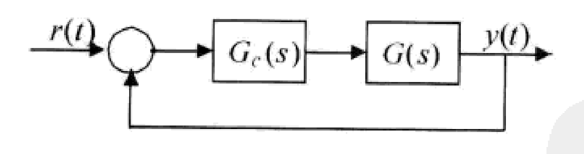


图 5-60 惯于控制系统框图

图中，被控对象 $G(s)=\dfrac{10}{300s+1}\mathrm{e}^{-150s}$，$G_c(s)$ 为控制器，试建立控制系统 Simulink 仿真模型，并利用 Ziegler-Nichols 法整定 PID 控制器参数。

表 5-1 中 PID 参数的 Ziegler-Nichols 经验整定公式可计算出 PID 控制器的初始参数值为 $K_p=0.24$，$T_i=350$，$T_d=350$。Simulink 求解器仿真终止时间设置为 2000s，其他参数取默认值。运行仿真，可得初步整定参数下系统单位阶跃响应，如图 5-61 所示。

从图 5-62 可以看出，系统超调量为稳态值的 30%，振荡次数为 3 次，峰值时间为 350s，基本符合要求。如果工作机构对系统超调量有严格要求，欲控制在 10%以内，则可以根据 PID 参数对控制系统性能的影响，在 Simulink 仿真模型里修改 K_p、T_i和 T_d 的数值。经

过反复调试，最后整定 PID 控制器参数为 $K_p=0.17$ $T_i=3.57$、$T_d=50$。运行仿真得到再次整定 PID 参数后的仿真结果，如图 5-63 所示。系统指标中，超条量 8%，峰值时间 420s。可以看出，修改 PID 参数后降低了系统的超调量，但也牺牲了系统的动态性能，满足工作机构对超调量的要求。

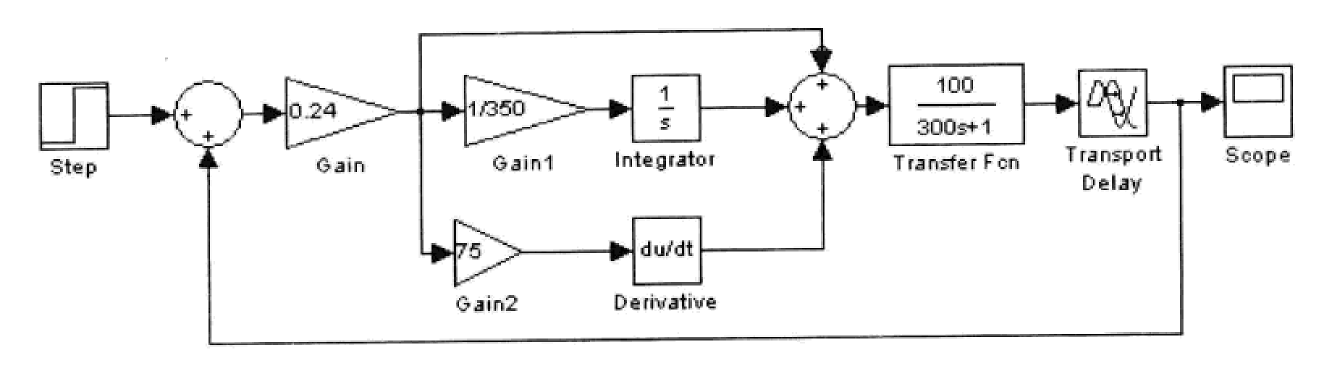


图 5-61　Simulink 仿真模型

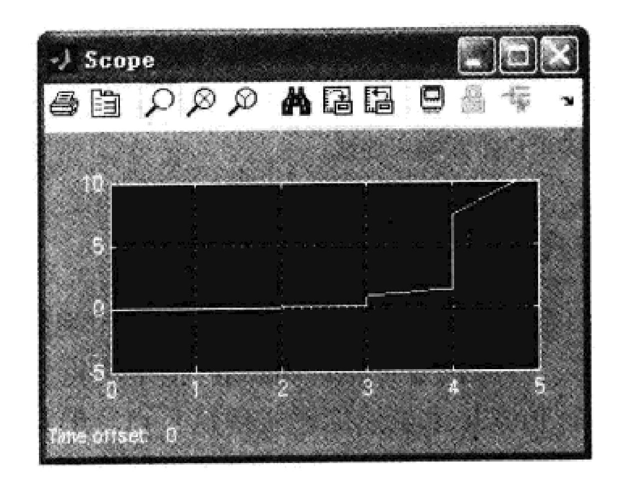


图 5-62　初步仿真结果

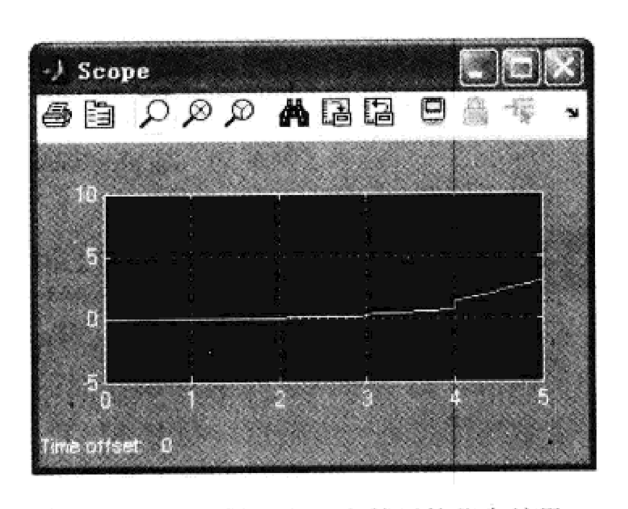


图 5-63　再次整定 PID 参数后的仿真结果

5.5.3 临界比例带法

临界比例带法适用于已知对象数学模型的场合，且被控对象是 3 阶或者 3 阶以上系统。在闭环控制系统中，将调节器置于纯比例作用下，从大到小改变调节器的比例带（或从小到大增加控制器的比例系数），得到等幅振荡的过程。此时的比例带称为临界比例带 δ_k，相临两个波峰的时间间隔称为等幅振荡周期 T_k。

临界比例带法整定 PID 控制器参数的基本步骤如下：

1）将 PID 控制器的积分时间常数置于无穷大（$T_i=\infty$），微分时间常数置零（$T_d=0$），比例带 δ 取为较大值，并将系统投入运行。

2）将比例 δ 逐渐减小（Simulink 仿真调试时可由小到大调节比例系数 K_p，等达到等幅振幅后根据此时的 K_p 再求取临界比例带 δ_k），得到等幅振荡过程，记下临界比例带 δ_k 和临界振荡周期 T_k。

3）根据 δ_k 和 T_k 的数值，按表 5-2 中的经验公式，计算出调节器的参数 δ、T_i 和 T_d。



表 5-2　临界比例带法整定 PID 控制器的参数公式

控制器类型	比例带 δ	积分时间 T_i	微分时间 T_d
P	$2\delta_k$	∞	0
PI	$2.2\delta_k$	$0.833T_k$	0
PID	$1.67\delta_k$	$0.5T_k$	$0.125T_k$

整定完调节器的参数后，按照先 P 后 I 最后 D 的操作规则将调节器投入运行。如果系统达不到期望指标，则可根据 PID 控制器的调节规律，进一步整定其参数。

【例 5-17】 已知单位负反馈控制系统开环传递函数如下：

$$G(s)=\frac{1}{s(s+2)(s+4)}$$

控制器为 PID 控制器，试采用临界比例带法整定 PID 参数，并求系统单位阶跃响应。

根据临界比例带法：第一步建立如图 5-64 所示的 Simulink 模型，并将 PID 控制器的积分、微分环节断开，置比例系数 Kp=1。

第二步：以 10 倍速度逐渐增大 Kp，当 Kp=100 时，系统输出发散。再以 1/2 调节时进行收敛。最后得到等幅振荡的 Kp=48。此时，临界比例带 δ_k=0.0208，临界振荡周期 T_k=2.2s。

第三步：根据 δ_k 和 T_k 的数值，按表 5-2 中的经验公式，计算出调节器的参数 δ=0.035、T_i=1.1s 和 T_d=0.275。

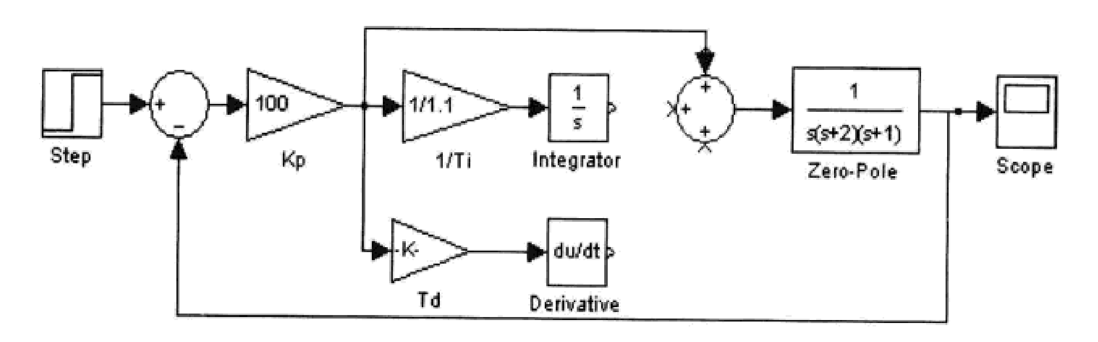


图 5-64　Simulink 模型

然后仿效将比例、积分、微分环节进行运动，可得如图 5-65 所示的仿真结果。

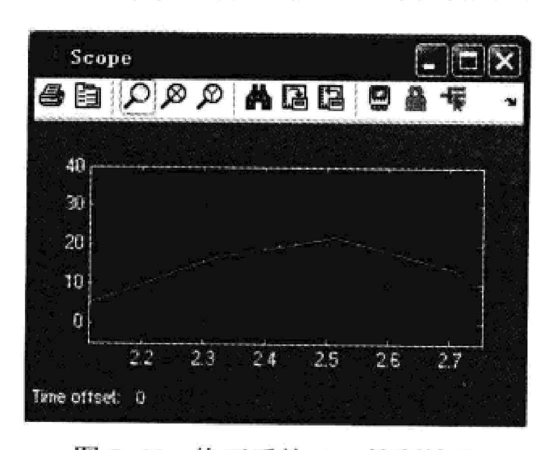


图 5-65　修正后的 PID 控制输出

第 6 章 Simulink 在电力系统的建模与仿真应用

电力系统一般由发电机、变压器、电力线路和电力负荷构成。电力系统的数学模型一般是由电力系统元器件的数学模型组合构成。MATLAB 为电力系统的建模提供了简洁的工具，通过电力系统的电路图绘制，可以自动生成数学模型。电路图模型具有良好的人机界面，便于进行简单的操作，省去了利用程序建立电力系统模型的繁琐步骤。利用这种方式构成的数学模型相对于控制系统中的微分方程模型、状态方程模型、传递函数模型具有更直观和实用的优点。另外，在电路图模型建立以后，在 MATLAB 软件中，提供了 powr2sys 函数作为短路模型的结构分析函数，可以利用 power2sys 函数将电力系统的电路图模型向状态方程模型和传递函数模型进行变换。

6.1 电力系统的模型分析

6.1.1 电力系统仿真工具箱介绍

在 Command Window（命令窗口）输入“powerlib”命令，系统弹出 SimPowerSystem5.1（电力系统仿真）工具箱，如图 6-1 所示。也可以通过 Simulink 库浏览器，单击“SimPowersystem”标题前的⊞节点，可逐级打开电力系统仿真工具箱的模块库。还可以通过 MATLAB 的开始菜单中单击“Start”开始菜单下的“Simulink”菜单下的“SimPowersystem”子菜单下的“Block Library”命令，系统弹出电力系统仿真工具箱模型窗口。在图 6-1 中，电力系统仿真工具箱包括 Electrical Sources（电源模块库）、Elements（电力元件模块库）、Prower Electronics（电力电子元器件模块库）、Machines（电动机模块库）、Measurements（测量模块库）、Application Libraries（应用模块库）、Extra Library（特别模块库）以及 powergui（电气系统仿真分析的图形用户接口）。

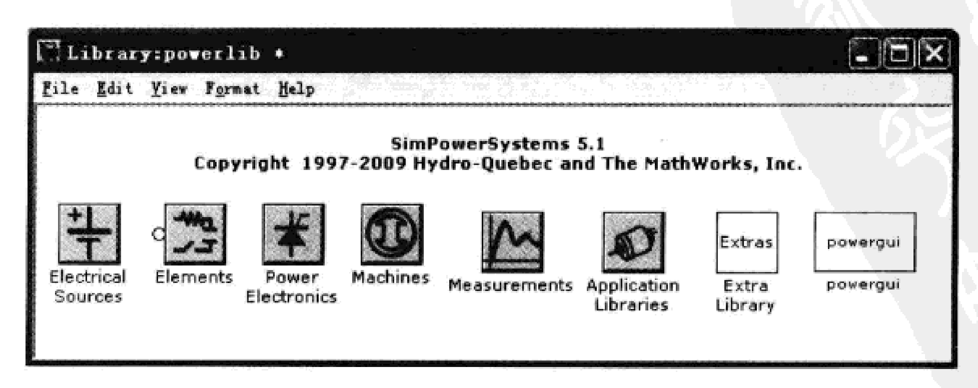


图 6-1 电力系统仿真工具箱界面

下面对各模块库进行简单介绍。

1. Electrical Sources 模块库

双击图 6-1 中的“Electrical Sources”模块，系统弹出如图 6-2 所示的电源模块库，包括 DC Voltage Source（直流电压源）、AC Voltage Source（交流电压源）、AC Current Source（交流电流源）、Controlled Voltage Source（可控电压源）、Controlled Current Source（可控电流源）、Three-Phase Source（三相电源）、Three-Phase Programmable Voltage Source（三相可编程电压源）以及 Battery（电池）模块。



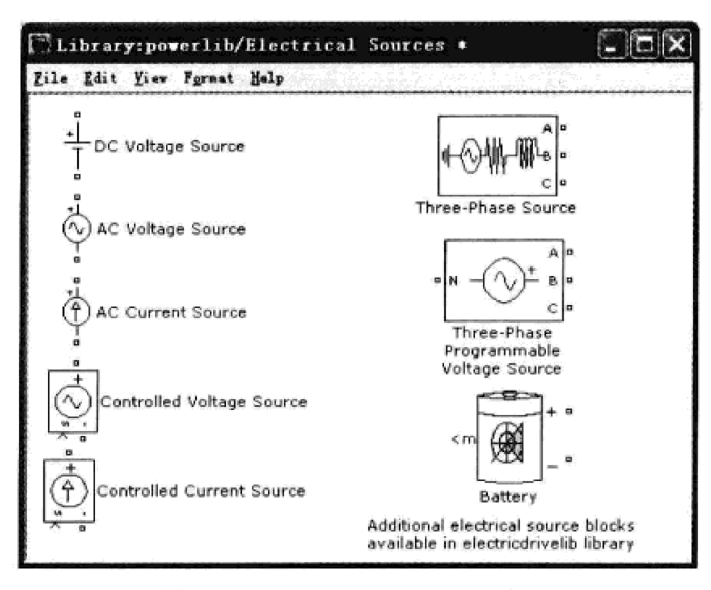


图 6-2 Electrical Sources 模块库

2. Elements 模块库

双击图 6-1 中的“Elements”模块，系统弹出如图 6-3 所示的电力元器件模块库（图中模块太多，未能展示完）。电力元器件模块库主要由 Elements（基本电力元器件）、Lines（电力连接线）、Circuit Breaker（电路开关）、Transformers（变压器）等组成，各部分又包含各类元器件，在这里不展开介绍。

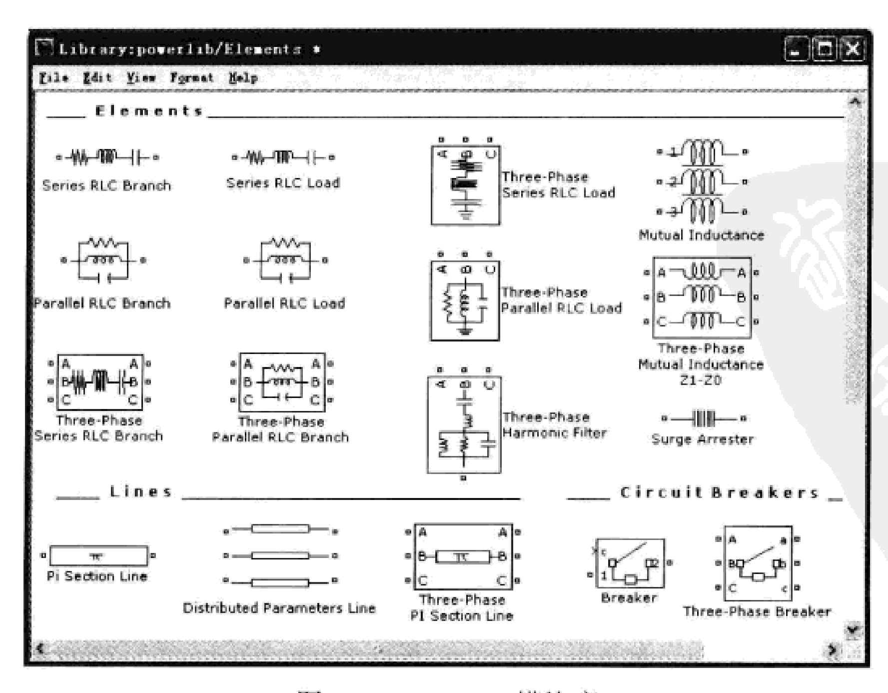


图 6-3 Elements 模块库

3. Power Electronics 模块库

双击图 6-1 中的“Power Electronics”模块，系统弹出如图 6-4 所示的电力电子元器件模块库，主要包括各类 Diode（二极管）、Thyristor（晶体管）、IGBT（场效应晶体管）等，此外还包含两个子模块库：Control blocks（控制模块库）和 Discrete Control blocks（离散控制模块库）。打开控制模块库，如图 6-5 所示，包括各种 Filters（滤波器模块）、Phase-Locked Loop（PLL）Systems（锁相环系统）、Pluse Generators（脉冲生成器）、Signal Generators（信号生成器）及其他模块库。

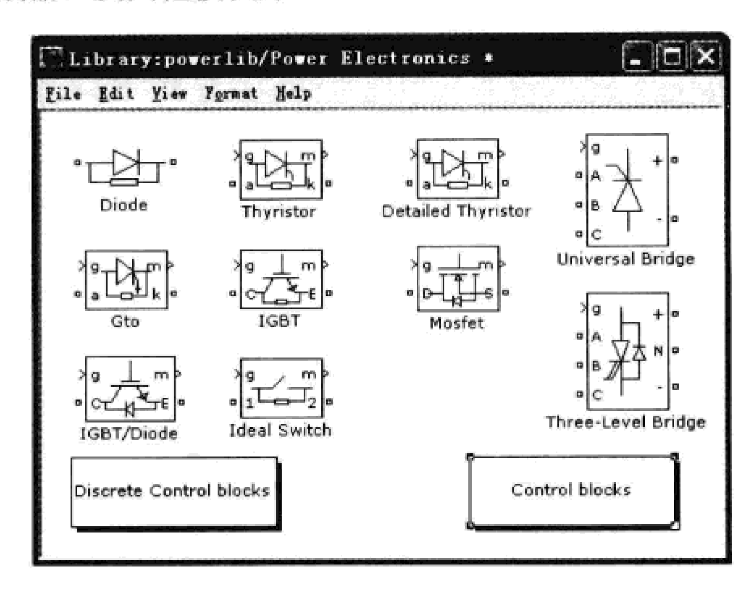


图 6-4 Power Electronics 模块库

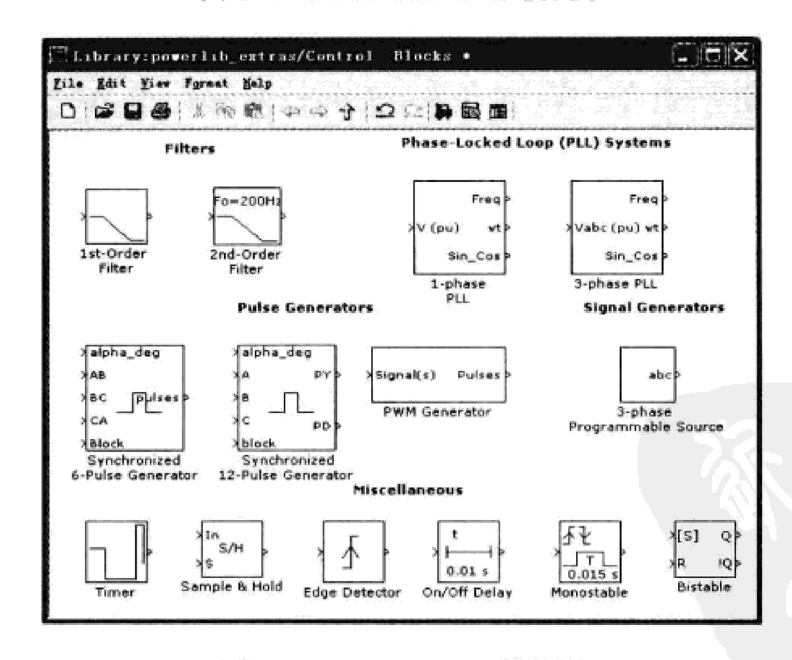


图 6-5 Control blocks 模块库

4. Machines 模块库

双击图 6-1 中的“Machines”模块，系统弹出如图 6-6 所示的电动机模块库。电动机模块库包括各类 Synchronous Machines（同步电动机）、Asynchronous Machines（异步电动机）、DC Machine（直流电动机）、Switched Reluctance Motors（绕线电动机）、Stepper Motor（步进电动机）及其他模块。



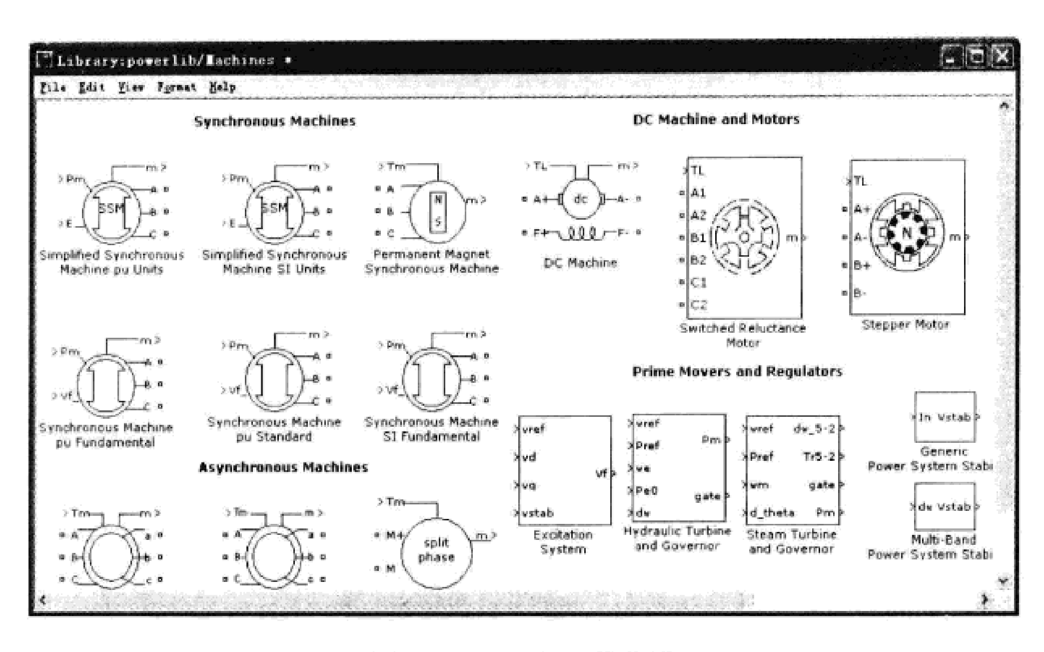


图 6-6 Machines 模块库

5. Measurements 模块库

打开 Measurements（测量）模块库，如图 6-7 所示，主要包括 Current Measurement（电流测量）模块、Voltage Measurement（电压测量）模块、Impedance Measurement（组抗测量）模块、Multimeter（多功能测量）模块、Three-Phase V-I Measurement（三相电压电流测量）模块。此外还包含了 3 个子系统模块库，分别是 Continuous Measurements（连续系统测量模块库）、Discrete Measurements（离散系统测量模块库）及 Phasor Measurements（相量测量模块库）。

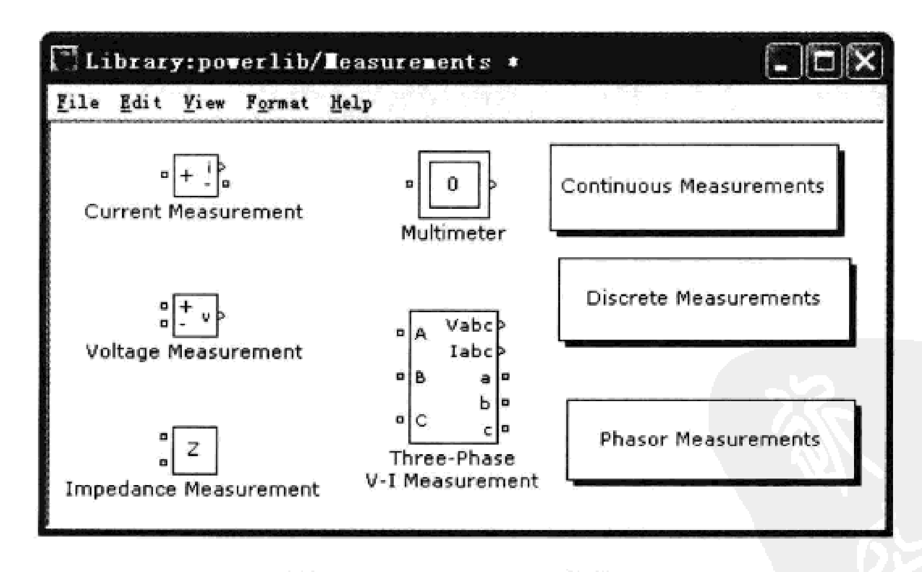


图 6-7 Measurements 模块库

6. Application Libraries 模块库

打开 Application Libraries（应用）模块库，如图 6-8 所示，主要包括 Electric Drives Library 模块库、Flexible AC Transmission Systems（FACTS）Library 模块库、Distributed Resources Library 模块库。打开电气传动子模块，如图 6-9 所示，包括 DC Drive（直流传动 Drive），如图 6-10 所示和 AC Drive（交流传动），如图 6-11 所示以及减速器模块。

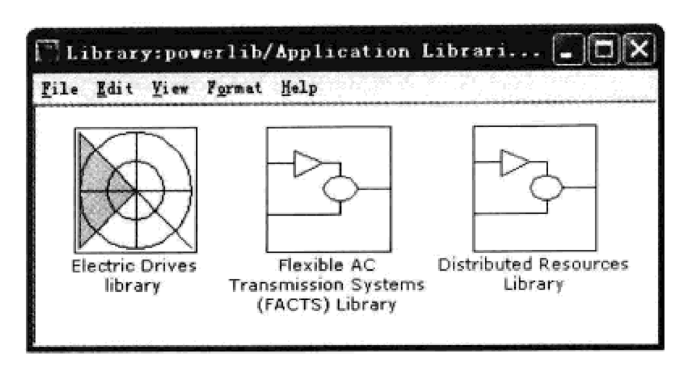


图 6-8 Application Libraries 模块库

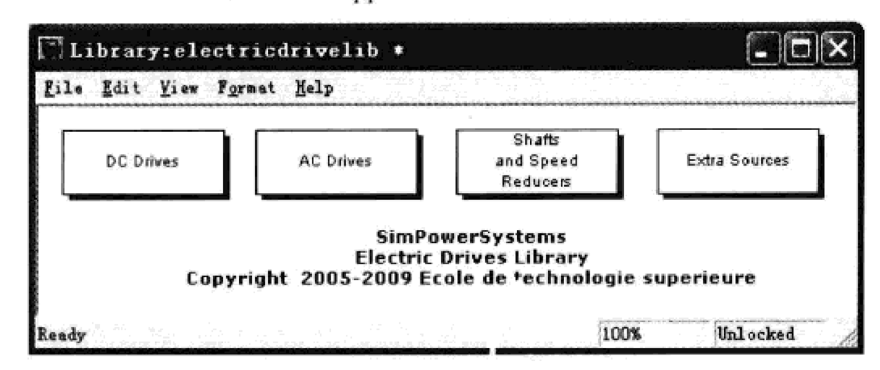


图 6-9 Electric Drives Library 模块库

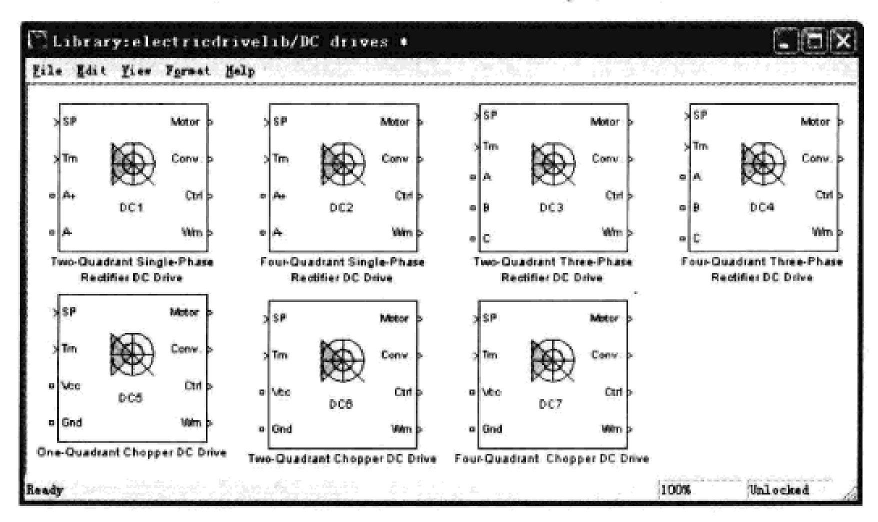


图 6-10 DC Drive 模块库

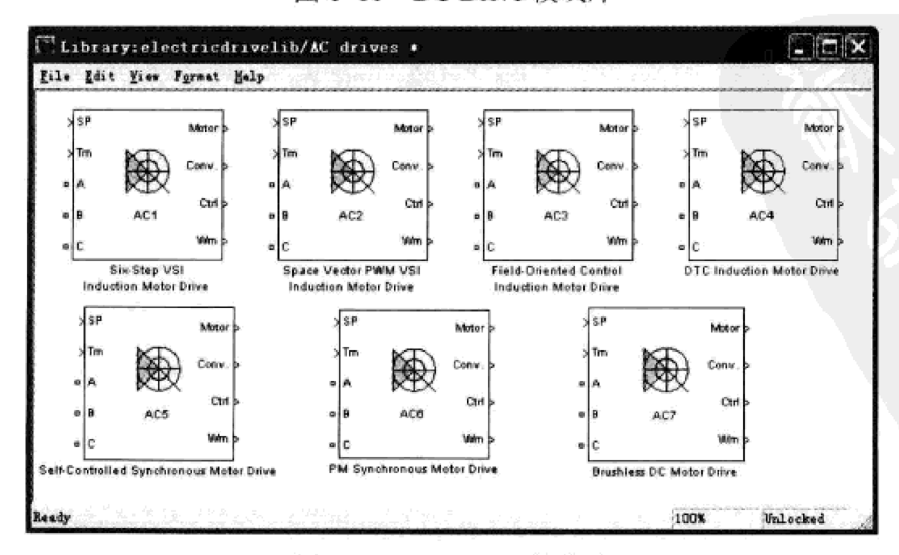


图 6-11 AC Drive 模块库

下面以一个简单的例子来介绍如何使用这些电气元器件。

【例 6-1】 交流电压源的叠加。设计两个单相交流电压源，叠加后作为电路的电源，分析电路首端电压的变化情况。

设计的交流电路，如图 6-12 所示，在此电路图中，交流电压源的幅值、频率、相位均不相同，可以通过仿真结果直接对各自电压源的输出和它们的叠加结果进行分析，这种分析方法简单、直接。



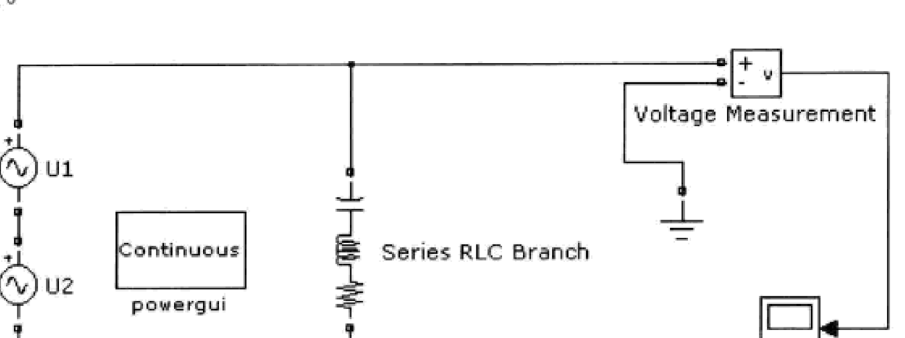


图 6-12　交流电流源的叠加电路

（1）电路图设计步骤

1）Electrical Sources 模块库选择交流电压源元器件，把它拖放到新建的电路图中。

① 将电压源元器件改名为 $u1$。

② 双击 AC Voltage Source（交流电压源元器件），对交流电压源元器件的参数进行如下设置，如图 6-13 所示。

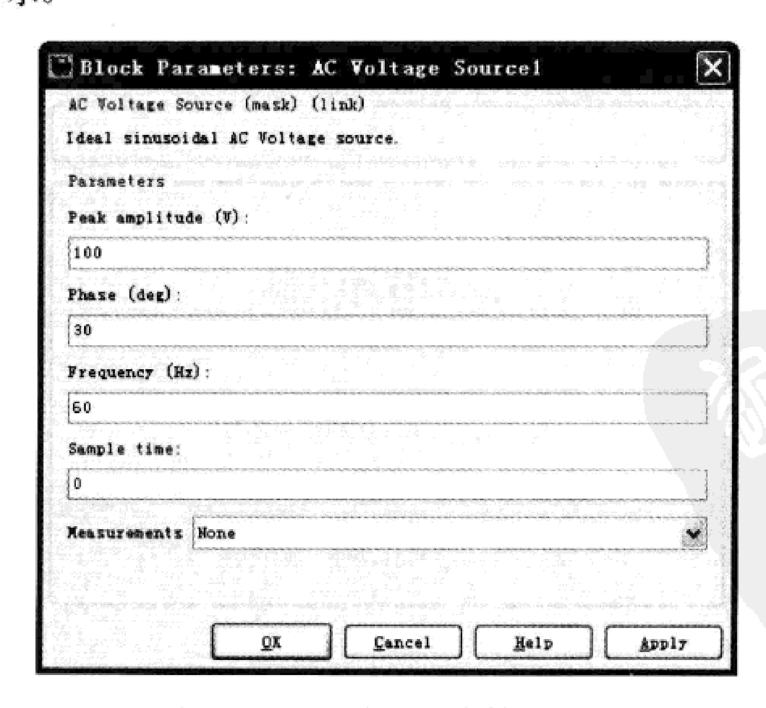


图 6-13　交流电源 u1 参数对话框

- Peak amplitude（峰值振幅）：100。
- Phase（初始相位）：30。

- Frequency（频率）：60。
- Sample time（采样时间）：0。
- Measurements（测量选项）：选择不测量电气量。

交流电压源 $u1$ 的表达式如下：

$$u1 = 100\sin\left(120\pi t + \frac{\pi}{6}\right)$$

③ 复制交流电压源元器件并改名为 $u2$。

④ 双击交流电压源元器件，对交流电压源元器件的参数进行如下设置，如图 6-14 所示。

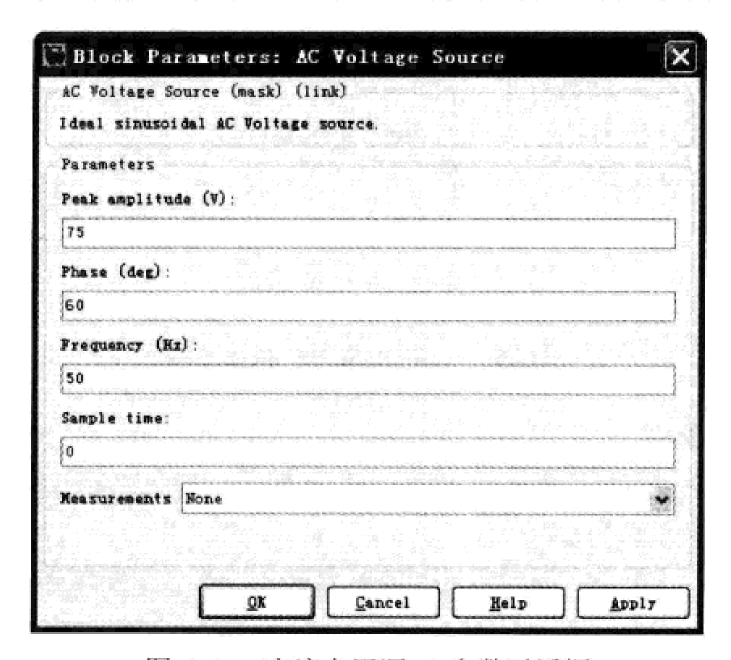


图 6-14　交流电压源 $u2$ 参数对话框

- Peak amplitude（峰值振幅）：75。
- Phase（初始相位）：60。
- Frequency（频率）：50。
- Sample time（采样时间）：0。
- Measurements（测量选项）：选择不测量电气量。

交流电压源 $u2$ 的表达式如下：

$$u2 = 75\sin\left(100\pi t + \frac{\pi}{3}\right)$$

2）从 Elements 模块库中选择串联 RLC 支路（Series RLC Branch）。

对串联 RLC 支路元器件的参数进行如下设置，如图 6-15 所示。

- Resistance（电阻）：200。
- Inductance L（电感）：100e-3。
- Capacitance（电容）：150e-3。
- Measurements（测量选项）：选择不测量电气量。

3）Measurements 模块库中选择 Voltage Measurement 元器件，拖放到电路图中。

4）在 Simulink Liberary（仿真元器件库）中选择示波器，拖放到电路图中。

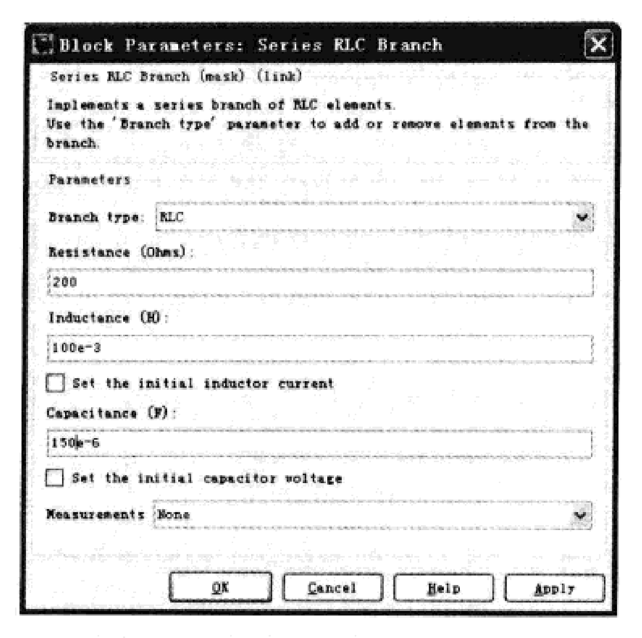


图 6-15 串联 RLC 支路参数对话框

5）从 Connectors（连接元器件库）中选择接地及相应的元器件进行合理的放置并连线。

对电路图进行连线完成时，就可以完成电路图的绘制。在接地时，如果提示颜色为红色，则表示在接线时出现了错误。

（2）仿真参数设置

在电路图菜单选项中，单击“Simulation”菜单下的“Configurtion Parameters”命令，即可对仿真参数对话框相应选项进行如下设置：

- Start time（开始时间）：0。
- Stop time（停止时间）：0.4。
- Type（求解程序类型）：Variable-step（可变步长），ode45（Dormand-Prince）。
- Max step size（最大步长）：auto（自动）。
- Min step size（最小步长）：auto（自动）。
- Initial step size（初始步长）：auto（自动）。
- Relative tolerance（相对容差）：1e-3。
- Absolute tolerance（绝对容差）：1e-6。

（3）仿真结果及分析

合理设置示波器参数后，激活仿真按钮，得到仿真结果如图 6-16 所示。

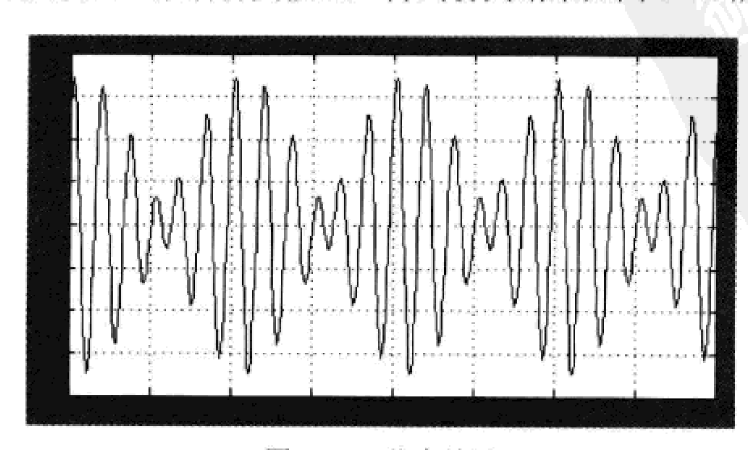


图 6-16 仿真结果

6.1.2 Park 变换分析

同步电动机是电力系统中的重要元器件，它实质上是由定子和转子两个部件组成。在研究同步电动机的数学模型时，假设定子三相绕组的结构完全相同，空间位置彼此相关 120°，转子铁心及绕组对极中心轴和极间轴完全对称。一般情况，在推导同步电动机的数学模型时应用的是用 abc 坐标系统表示的电压和磁链方程。abc 三轴就是定子三相绕组的中心轴线。定子三组绕组中的电流分别表示如下：

$$
\begin{aligned}
i_a &= I_m \cos(\omega_s t + \gamma_0) = I_m \cos\gamma \\
i_b &= I_m \cos(\gamma - 120°) \\
i_c &= I_m \cos(\gamma + 120°)
\end{aligned}
$$

利用该坐标系统建立同步电动机的电压和磁链方程时非常容易理解，但是所建立的方程为变系数的微分方程，它们的求解非常困难。为了克服这个困难，最简单有效的方法时将定子 abc 三相绕组的磁链和电压方程用一组新的变量替换，这样使方程更易于求解。变量变换又称做坐标变换，最常用的坐标变换是 Park 变换。Park 变换是将 abc 坐标系统下的 i_a、i_b、i_c 表示成 dq0 坐标系统下的 i_d、i_q、i_0。d 轴为转子中心线，称做纵轴或直轴；q 轴为转子极间轴，称做横轴或交轴，按转子旋转方向，q 轴比 d 轴超前 90°；0 坐标轴是抽象的。这样变换后电流的表示方式如下：

$$
\begin{aligned}
i &= i_d + i_q \\
i_a &= i_{ad} + i_{aq} \\
i_b &= i_{bd} + i_{bq} \\
i_c &= i_{cd} + i_{cq}
\end{aligned}
$$

1）abc 坐标系统变换为 dq0 坐标系统的变换公式如下：

$$
\begin{pmatrix} i_d \\ i_q \\ i_0 \end{pmatrix} = \frac{2}{3}\begin{pmatrix} \cos\theta & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta + \frac{2\pi}{3}\right) \\ -\sin\theta & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta + \frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}\begin{pmatrix} i_a \\ i_b \\ i_c \end{pmatrix}
$$

在 MATLAB 中，使用 abc_to_dq0 Transformation（abc 坐标系统转换为 dq0 坐标系统）元器件可以实现这种变换。abc_to_dq0 Transformation 在 PowerLib（电力系统元器件库）中的 Extras（附加元器件）下的 Measurements（测量元器件）中。其元器件及对话框，如图 6-17 所示。

2）dq0 坐标系统变换为 abc 坐标系统的变换公式如下：

$$
\begin{pmatrix} i_a \\ i_b \\ i_c \end{pmatrix} = \frac{2}{3}\begin{pmatrix} \cos\theta & -\sin\theta & 1 \\ \cos\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{2\pi}{3}\right) & 1 \\ \cos\left(\theta + \frac{2\pi}{3}\right) & -\sin\left(\theta + \frac{2\pi}{3}\right) & 1 \end{pmatrix}\begin{pmatrix} i_d \\ i_q \\ i_0 \end{pmatrix}
$$

在 MATLAB 中，使用 dq0_to_abc Transformation（dq0 坐标系统转换为 abc 坐标系统）软件可以实现这种变换。该元器件也在 PowerLib（电力系统元器件库）中的 Extras（附加元器件）下的 Measurements（测量元器件）中。其元器件及对话框如图 6-18 所示。



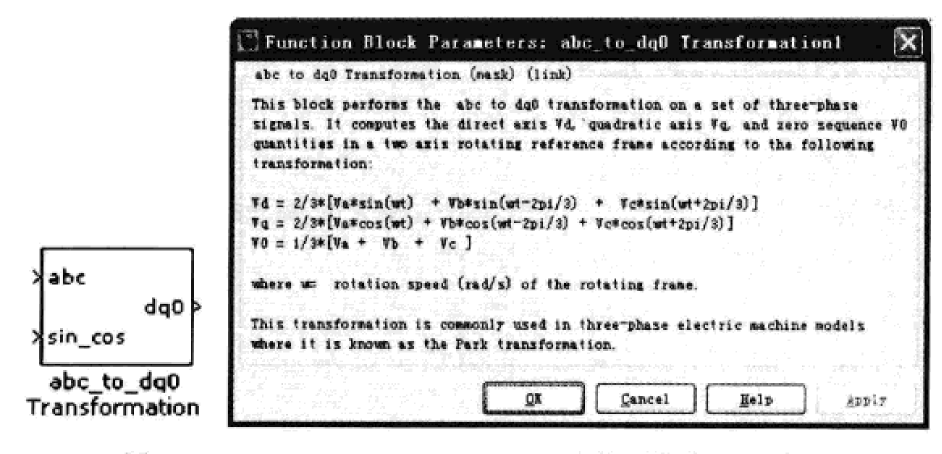


图 6-17　abc_to_dq0 Transformation 变换元器件及对话框

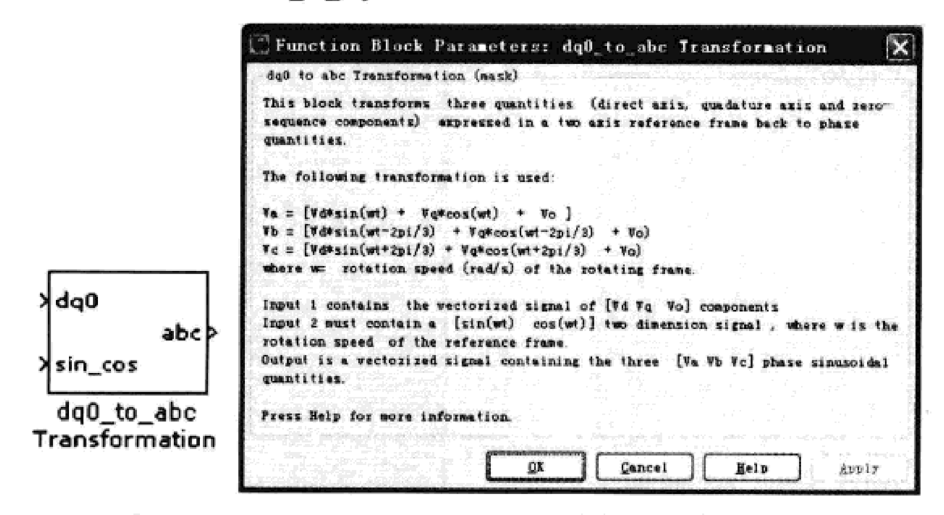


图 6-18　dq0_to_abc Transformation 变换元器件及对话框

下面给出一个简单的示例来介绍坐标变换。

【例 6-2】　坐标变换。

将给出的电路图用 park 变换从 abc 坐标系统转换为 dq0 坐标系。

（1）电路图设计

按照前面介绍的方法建立电路图模型，其建立的 Simulink 模型图如图 6-19 所示。

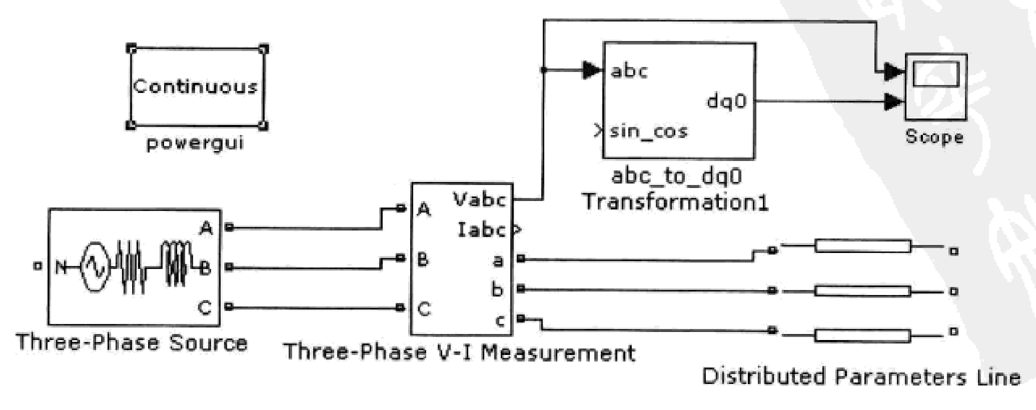


图 6-19　电路模型

（2）仿真参数设置

设置三相交流电压源的参数，如图 6-20 所示。

设置三相分布参数等值电路元器件参数，如图 6-21 所示。

在电路图菜单选项中，单击“Simulation”菜单下的“Configurtion Parameters”命令，即可对仿真参数对话框相应选项进行如下设置：

- Start time（开始时间）：0。
- Stop time（停止时间）：0.1。
- Type（求解程序类型）：Variable-step（可变步长），ode15s（stiff/NDF）。
- Max step size（最大步长）：auto（自动）。
- Min step size（最小步长）：auto（自动）。
- Initial step size（初始步长）：auto（自动）。
- Relative tolerance（相对容差）：1e-3。
- Absolute tolerance（绝对容差）：auto。

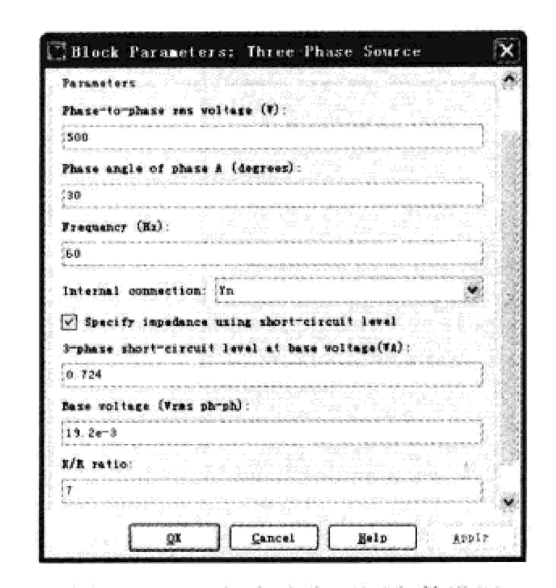


图 6-20　三相交流电压源参数设置

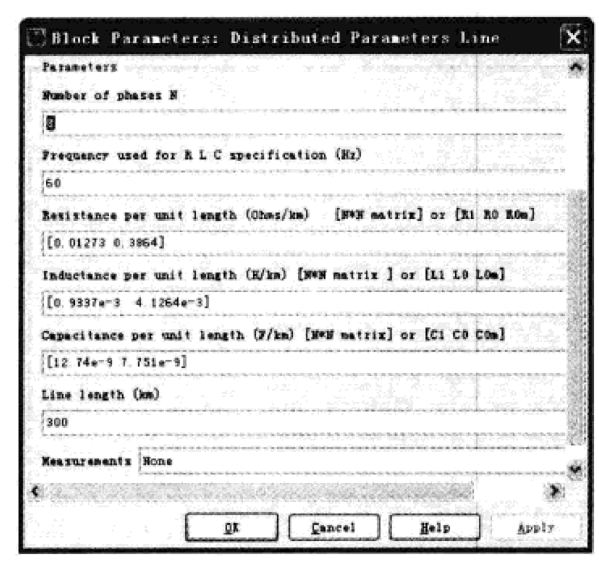


图 6-21　三相分布参数等值电路元器件参数设置

（3）仿真结果及分析

参数设置完成后，进行仿真，仿真结果如图 6-22 所示。其中，上面的曲线为三相坐标系统下的电压波形，下面的曲线为 dq0 坐标系统下的电压曲线。

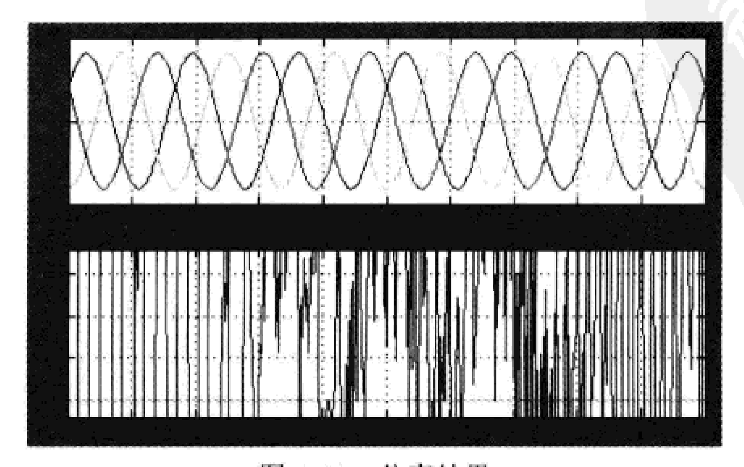


图 6-22　仿真结果

6.1.3 三相桥式全控制电流电路分析

根据三相桥式全控整原理，可建立如图 6-23 所示的三相桥式全控整流电路的 Simulink 仿真模型。图中三相电源 Usa、Usb、Use 的幅值设置为 311V、频率 50Hz，相位分别为 0°、-12°、120°；串联 RCL 元件选择电阻；阻值设置为 10Ω；仿真求解器选择为 ode15s，最大补偿设置为 0.01s；整流控制角 alpha 设置为 0。运行仿真，得到如图 6-24a 所示的整流电压输出及如图 6-25a 所示的同步六脉冲波形。改变触发角 alpha 为 30，运行仿真，可得到如图 6-24b 所示的整流电压输出及如图 6-25b 所示的同步六脉冲波形。



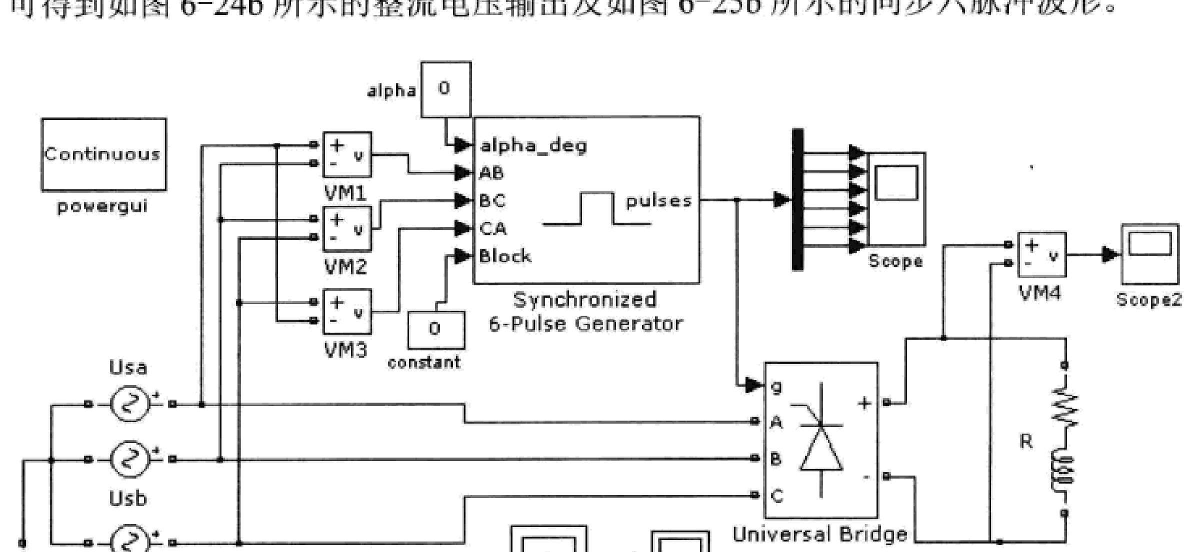
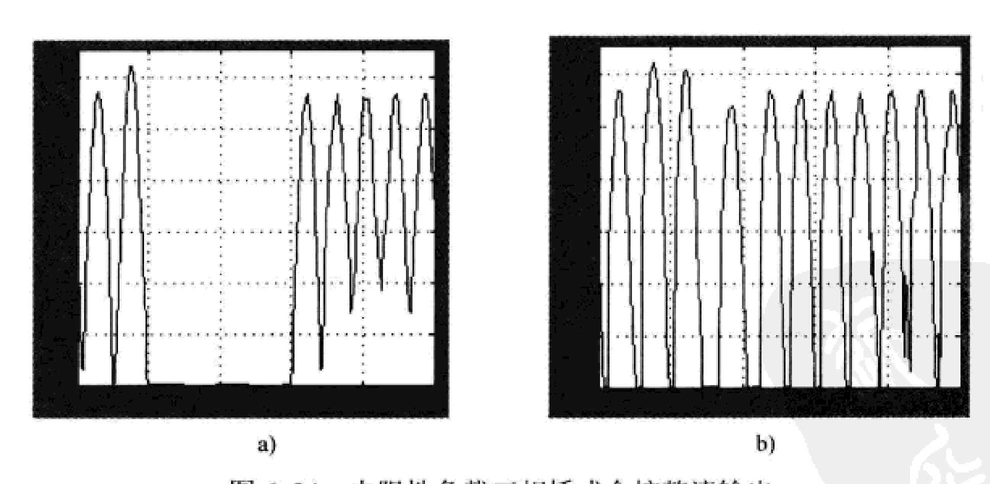


图 6-23 三相桥全控整流电路



a) b)

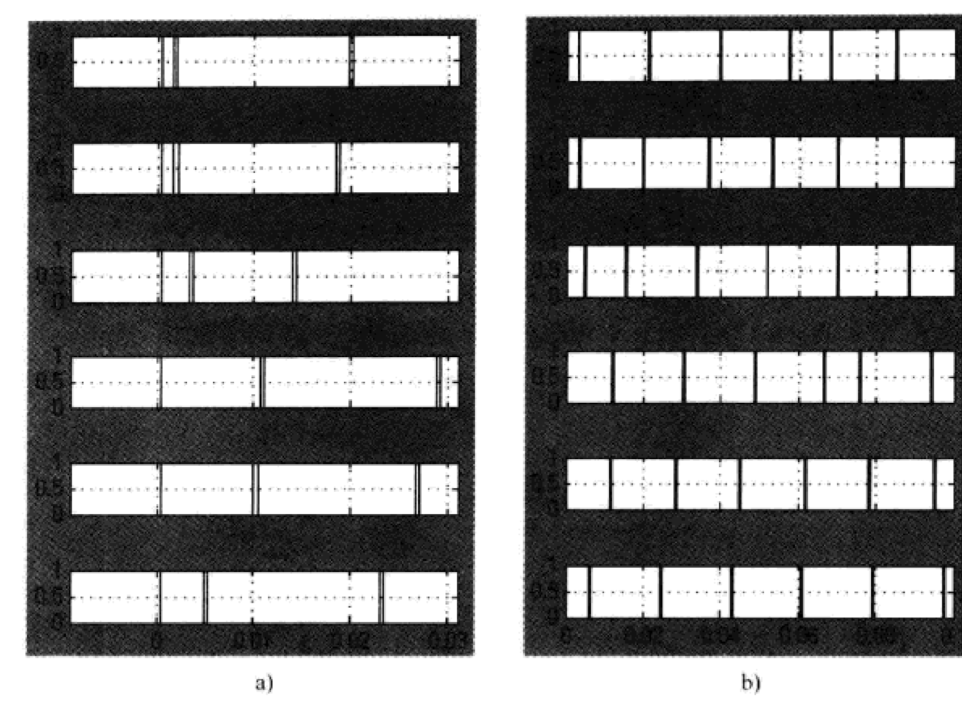
图 6-24 电阻性负载三相桥式全控整流输出

a) 电阻性负载触发角 0° 时整流输出 b) 电阻性负载触发角 30° 时整流输出

将负载串联 RCL 元件选择为 RL 的感性负载，R=10Ω，L=1H，整流输出波形如图 6-26 所示。

再将电感性负载并接一个滤波电容 C=0.05F，其参数设置如图 6-27 所示。

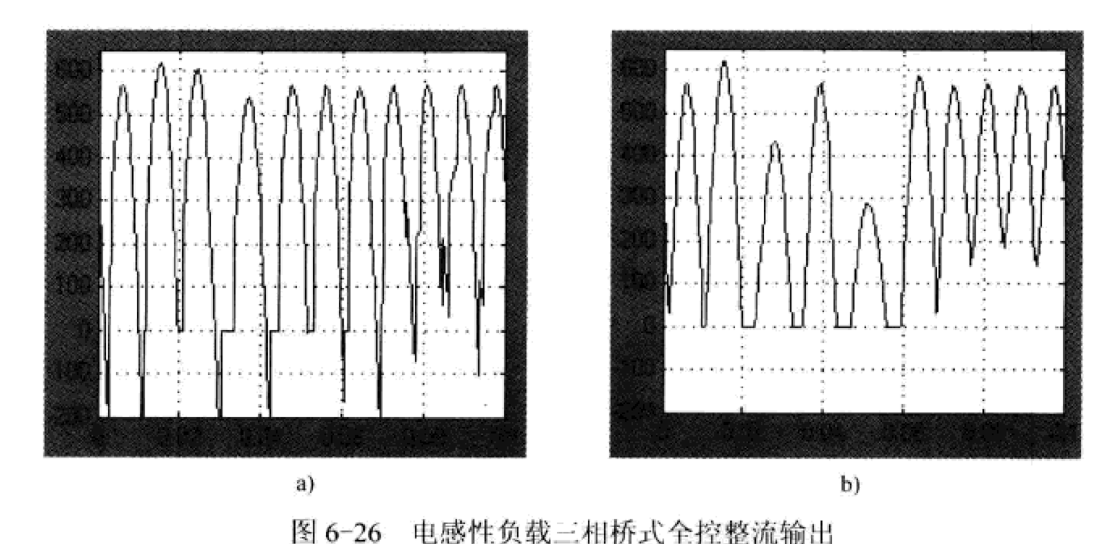
此时三相桥式全控整流电路如图 6-28 所示。限流电阻 R=100Ω，输出波形如图 6-29 所示。



a)　　　　　　　　　　　　　　b)

图 6-25　三相桥式全控整流同步六脉冲

a) 触发角 0° 时同步六脉冲　b) 触发角 30° 时同步六脉冲



a)　　　　　　　　　　　　　　b)

图 6-26　电感性负载三相桥式全控整流输出

a) 电感性负载触发角为 0° 时整流输出　b) 电感性负载触发角为 30° 时整流输出

Block Parameters: C

Series RLC Branch (mask) (link)

Implements a series branch of RLC elements.
Use the 'Branch type' parameter to add or remove elements from the branch.

Parameters

Branch type: C

Capacitance (

0.05

Set the in

Measurements

RLC
R
L
C
RL
RC
LC
Open circuit

OK　Cancel　Help　Apply

图 6-27　滤波电容参数设置

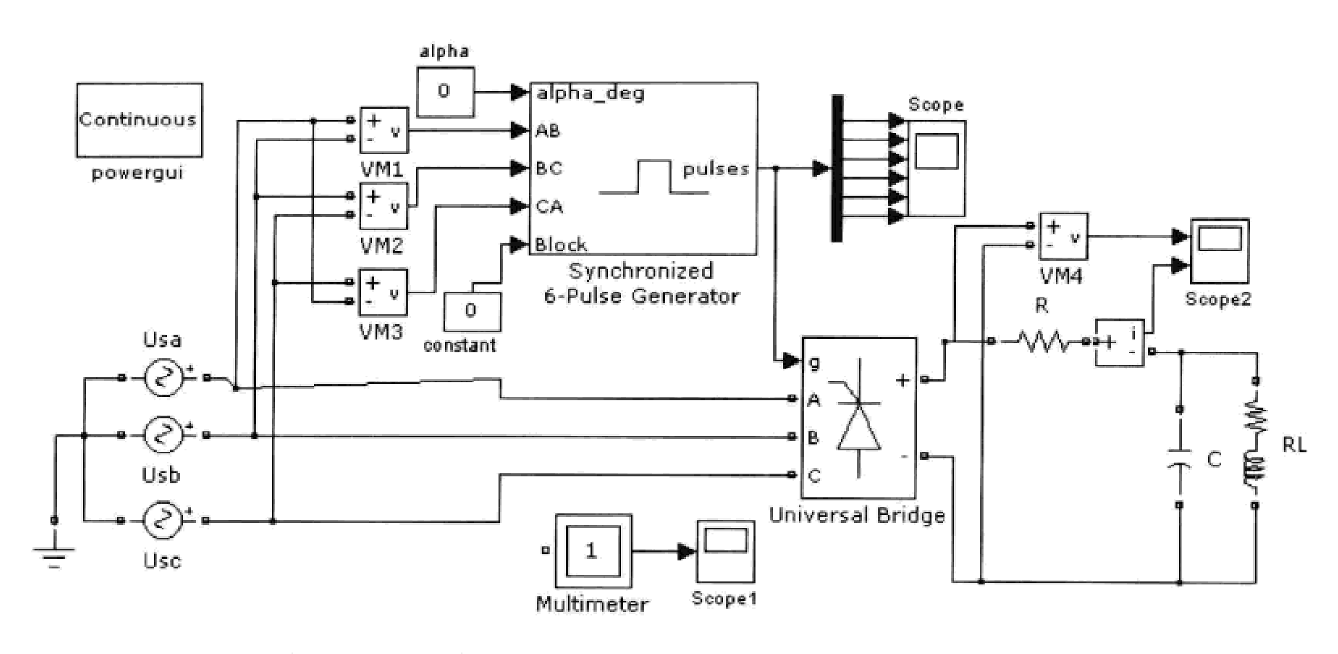
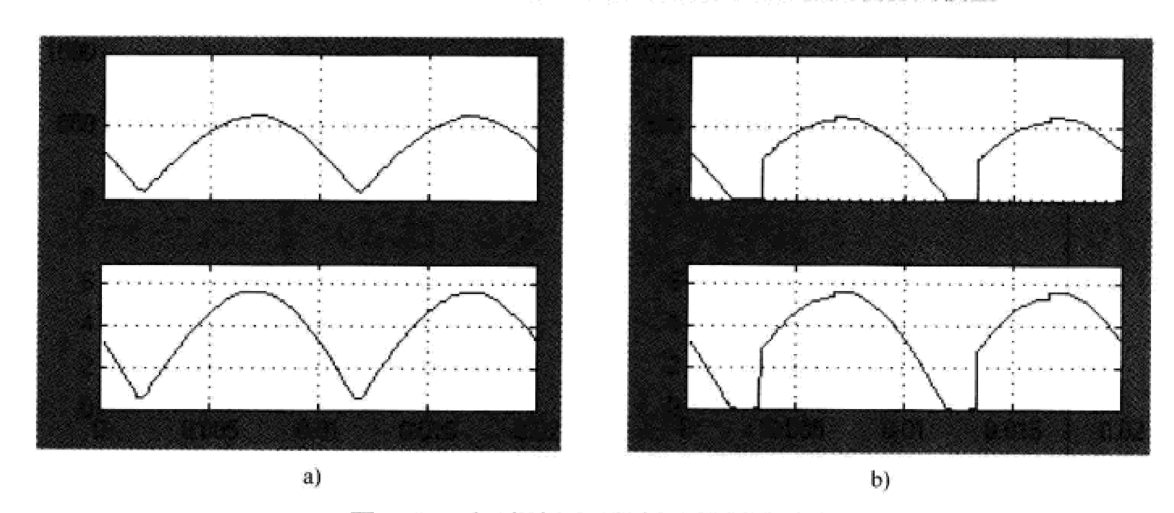


图 6-28　三相桥全控整流电流加输出滤波的电感性负载仿真模型



a)　　　　b)

图 6-29　加滤波后三相桥全控整流输出

a) 加滤波后触发角为 0° 时整流输出　b) 加滤波后触发角为 30° 时整流输出



经过上述三相桥全控整流电路的建模与仿真，可以得出使用 SimPowersystem 工具箱建模与仿真中的几点注意事项。

1）模块之间的连接线有两种类型：一种是 Simulink 里带方向的信号线；另一种是无方向可任意在同类线之间连接的“电力线”，而且它们连接在电力系统模块具有“。”的节点上（称为电力连接点）。两类连接线之间不能直接相连，这也就引出了电力测量模块，它完成将电力系统信号进行测量转换为 Simulink 模块能接收的数字信号。虽然 SimPowersystem 也是建立在 Simulink 平台上的计算机数字仿真技术，但属于一种“半模拟式”的计算机仿真技术。在理解上，我们可称其信号为“电力信号”，而 Simulink 仿真模型里的信号可称为“数字信号”。

2）在建立了 Simulink 仿真模型后，必须添加电力系统电路分析的图形用户接口“Powergui”模块，而且不能改变其名称，这是电力系统仿真必须添加的模块。如果用户忘记添加，则运行仿真时系统将提醒用户。

3）电力系统属于刚性系统，一般选择适合于刚性系统的求解器算法，如 ode15s、

ode23s、ode23t 等，并终止仿真。

4）在电力系统仿真建模时，至少要添加一个测量模块（任意一个），否则将提示警告信息，并终止仿真。

5）当系统比较复杂，需要观测的量较多，在使用电压、电流测量模块进行观测时，仿真模型连接线较多，影响模型的可读性。此时可以使用多测量模块（Multimeter），该模块的最大优点是示波器与被测信号间不需要连接线，而且可以测量多个量。其使用方法是：首先在系统模型中双击信号输出模块，在弹出的参数设置对话框中设置需要观测的信号。以电源模块为例，其参数设置对话框如图 6-30 所示，在“Measurments”下拉列表框中选择模块需要测量的的量为“Voltage”。同理可设置其他需要测量的信号输出模块，然后双击“Multimeter”模块，系统弹出参数设置对话框如图 6-31 所示。所有已被设置测量的量将在图 6-31 所示对话框的左边显示，通过中间区域的按钮可对所需信号进行选择、删除、移动等操作。如图 6-31 所示的设置，在图 6-23 模型窗口的工具栏中单击“▶”按钮，运行仿真，可得如图 6-32 所示的测量结果。

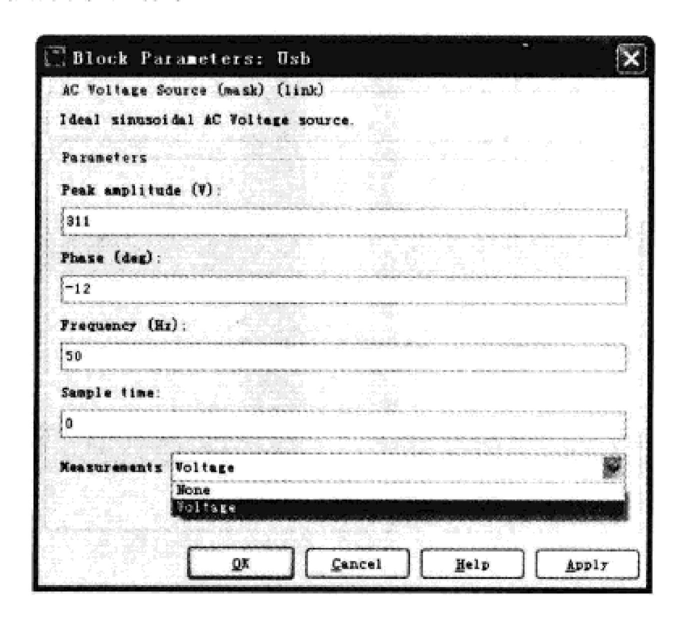


图 6-30　电源模块属性设置对话框

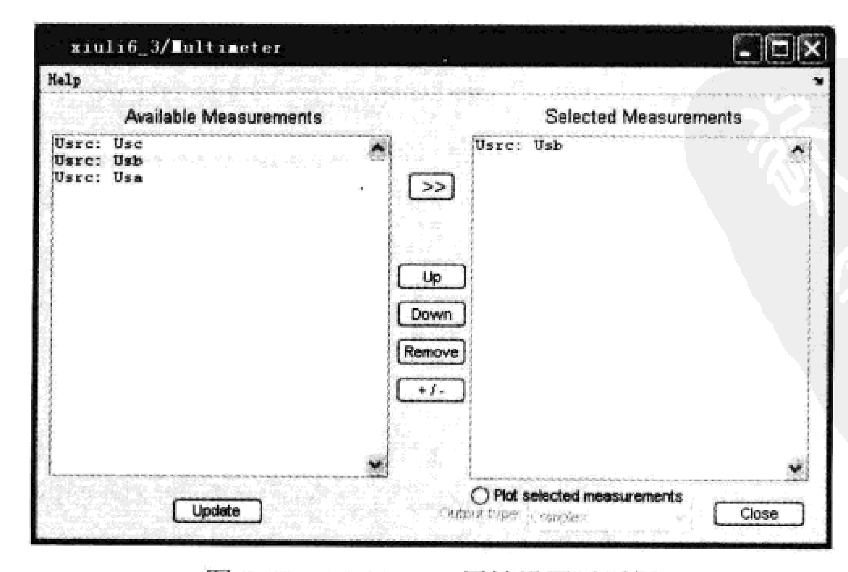


图 6-31　Multimeter 属性设置对话框

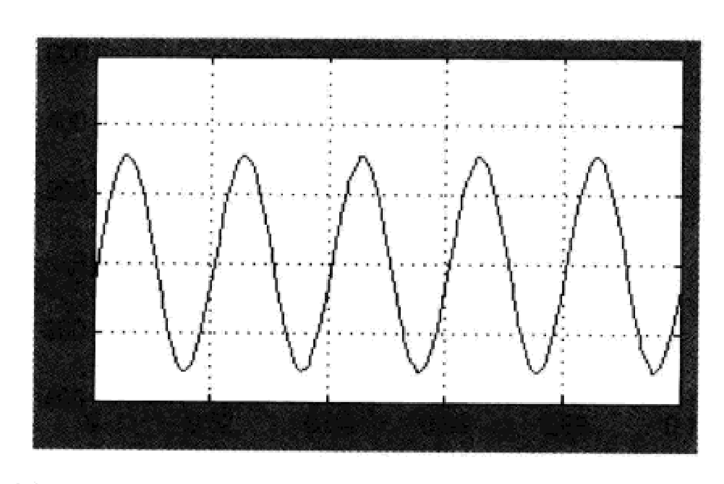


图 6-32 Multimeter 观测的电源电压和整流输出电流信号



6.2 直流调速系统的仿真分析

6.2.1 直流调整速系统控制方法分析

直流调速系统是以直流电动机为控制对象，主要以调节电动机的转速为目的，来满足实际生产过程的需要，而组成的控制装置的总称。因此在直流控制系统所研究的内容便是如何控制和调节电动机转速，得到理想的系统的动态和静态性能指标。

电枢回路的微分方程式为：

$$e_d = i_d R_d + L_d \frac{\mathrm{d}i_d}{\mathrm{d}t} = u_d \tag{6-1}$$

式中，e_d 为电动机电枢反电动势，单位为 V；R_d 为电动机电枢电阻，单位为Ω；i_d 为电动机电枢回路电流，单位为 A。

由于电机产生的反电动势为：

$$e_d = C_e n \tag{6-2}$$

式中，C_e 为电动机电势常数，单位为 $\mathrm{V/r \cdot min^{-1}}$

根据式（6-1）电动机的转速表达式为：

$$n = \frac{u_d - i_d R_d}{C_e} \tag{6-3}$$

由式（6-3）可以看出，调节直流电动机的转速有如下 3 种方法：

1）调节电枢电压调速。

2）改变电动机励磁调速。

3）改变电枢回路电阻调速。

改变电动机励磁调速方案一般是在基速以上实现调速，它的调速范围较窄，在实际应用较少；改变电枢回路电阻调速方案采用逐段增加或切除串入电动机回路中的电阻来实现的，导致部分电能无谓消耗在电阻的发热，而且不能实现无级调速。为了说明后两种调速方案的

实现，在这里以改变电枢回路电阻调速来说明在 MATLAB 中仿真实现。

1．模块使用

（1）DC-Motor（直流电动机）

直流电动机模块选自 SimPowerSystems 工具箱中的 machine 库里的 DC machine 模块，其模块及相关参数设置如图 6-33 所示。

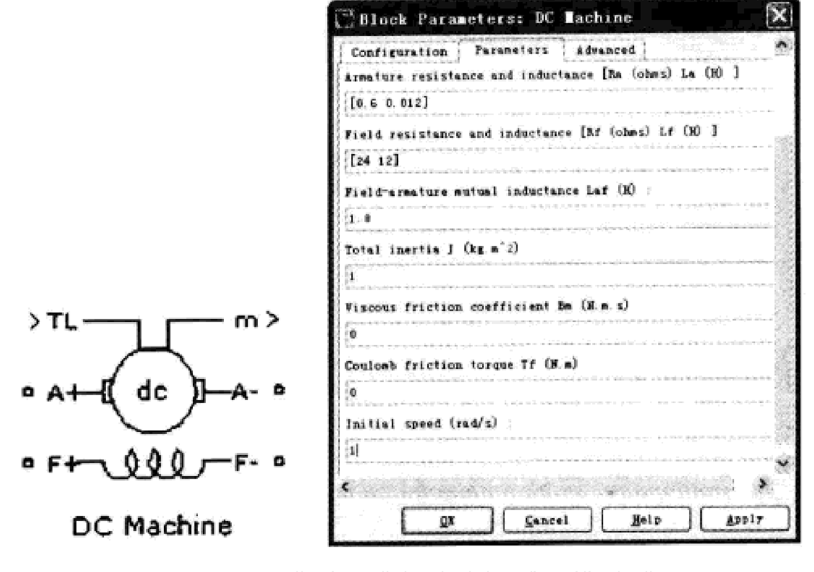


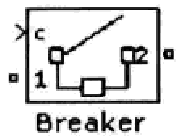
图 6-33　直流电动机模块及其参数设置

（2）直流电压源（E、E_f）

模块选自 SimPowerSystems 工具箱中的 Electrical Sources 库里的 DC Voltage source 模块。直流电压 E 为直流电动机的电枢回路电压，直流电压 E_f 直流电动机的励磁电压，Amplitude（两者参数）设置为 240。

（3）Breaker（断路器）

断路器选自 SimPowerSystems 工具箱中的 Elements 库里的 Breaker 模块，这里使用两只断路器，分别对电枢回路的串联电阻进行两级切除，从而实现改变电枢回路电阻调速过程。断路器模块及其参数设置，如图 6-34 所示。



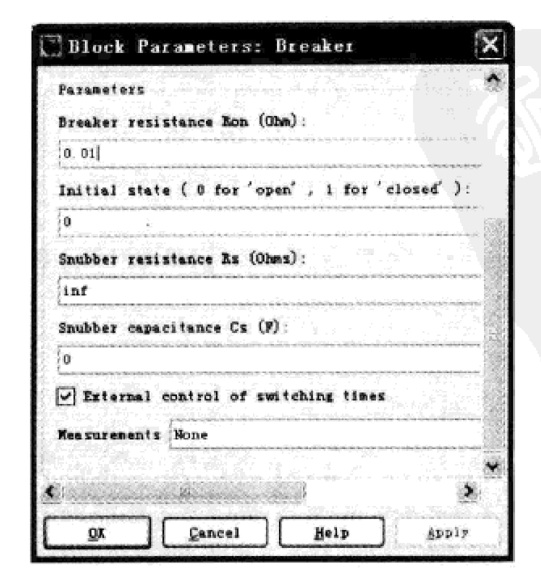


图 6-34　断路器模块及其参数设置

（4）调速电阻

调速电阻（*R*）选自 SimPowerSystems 工具箱中的 Elements 库里的 Serices RLC Branch 模块，为了说明原理，两只调速电阻都选择 20Ω，参数设置如图 6-35 所示。



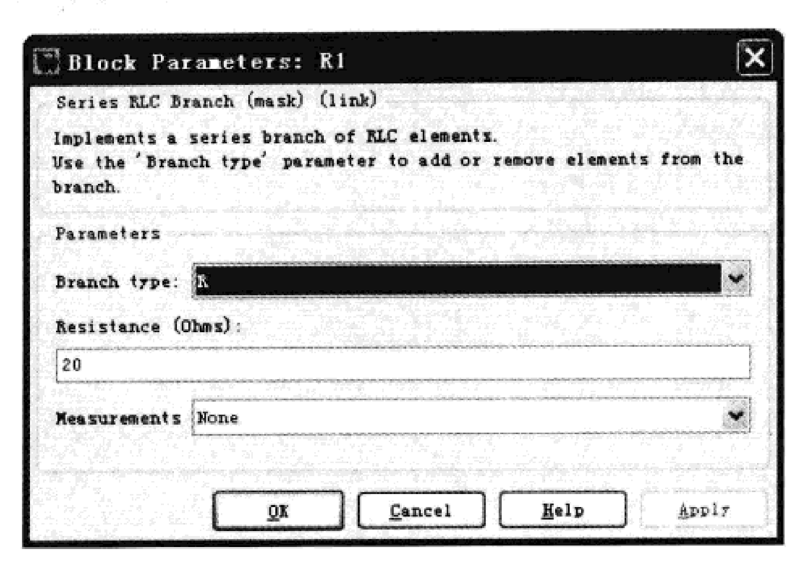


图 6-35　调速电阻参数设置

（5）Step（断路器控制）信号

断路器通断控制采用阶跃信号与模块的控制端连接实现，直流电动机的加速点分别设置在 5s 和 10s 时刻，因此将阶跃信号的跳变点时间分别为 5s 和 10s。

（6）其他相关模块

其他模块包括比例模块、输入型接地点，输出型接地点，两只 T 型连接器、信号分离器以及相关的示波器。

合理的调整各个模块的位置，再完成各个模块的信号连接，连接好的模型如图 6-36 所示。

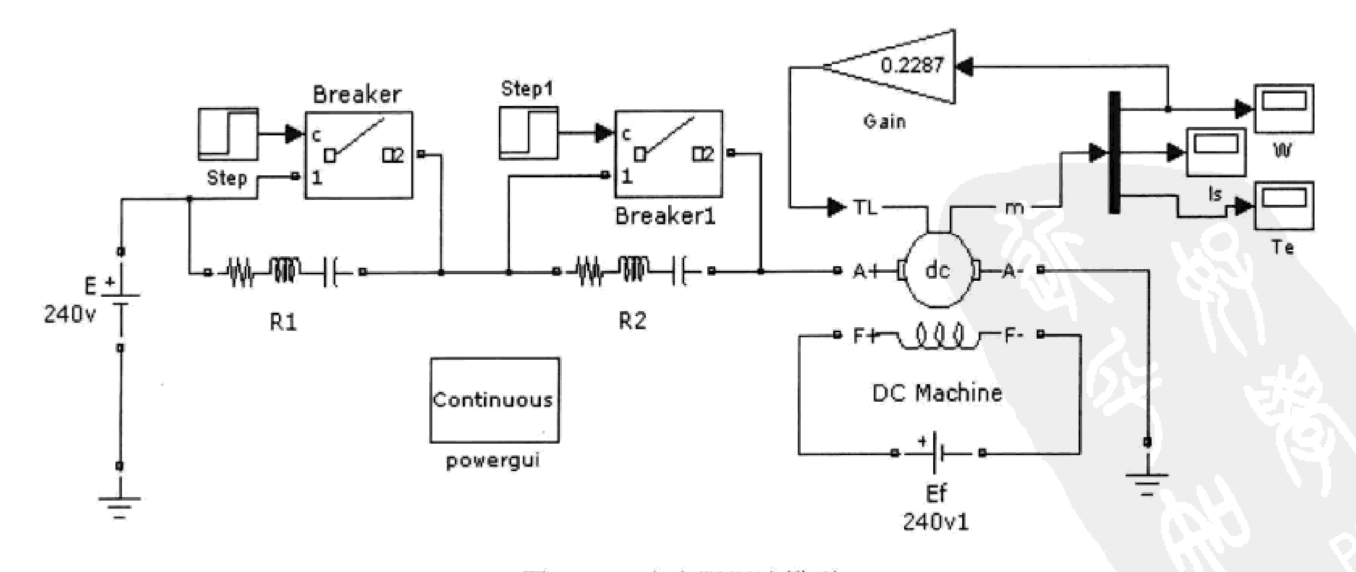


图 6-36　串电阻调速模型

2. 仿真参数设置

对本系统仿真参数设置如下：

- Start time（开始时间）：0。
- Stop time（停止时间）：15。
- Type（求解程序类型）：Variable-step（可变步长），ode15s（stiff/NDF）。
- Max step size（最大步长）：auto（自动）。
- Min step size（最小步长）：auto（自动）。
- Initial step size（初始步长）：auto（自动）。
- Relative tolerance（相对容差）：1e-3。
- Absolute tolerance（绝对容差）：auto。

3. 仿真结果

设置或修改仿真参数后，下面可以进行仿真运行，单击“运行”按钮，激活对应的示波器查看系统运行参数。电动机的转速 w（rad/s）如图 6-37 所示。可以看出在电动机稳定运行之后在 5s 和 10s 时刻切除电阻 1 和电阻 2 实现的电动机转速的调节。当然也可以借助此方法实现直流电动机的串阻启动。电枢电流 I_a（A）和电磁转矩 T_e（N·m）分别如图 6-38 和图 6-39 所示。

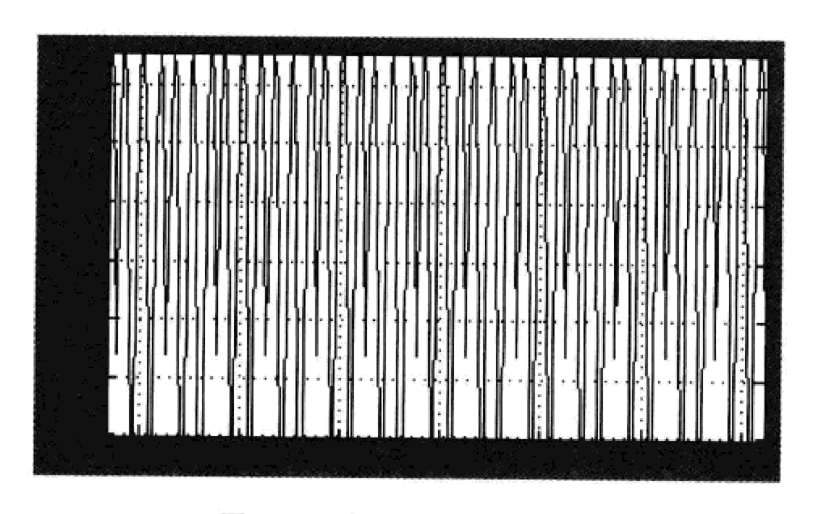


图 6-37　电动机的转速波形图

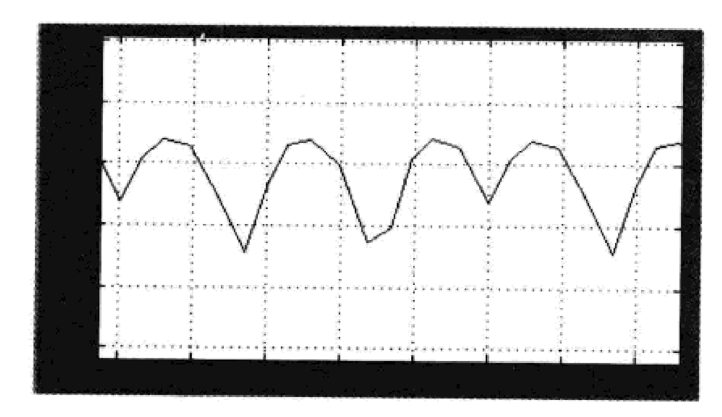


图 6-38　电枢电流波形图

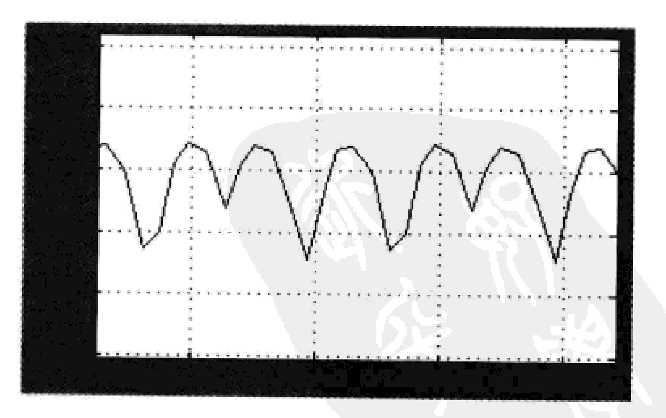


图 6-39　电磁转矩波形图

调节电枢电压调速方法优点很多，在实际工程中应用最多。

6.2.2 开环直流调速控制系统与仿真分析

根据系统的结构形式的不同将电动机控制系统分成开环直流调速控制系统和闭环直流调

速控制系统，下面就电动机控制系统的两种形式分别介绍，并进行仿真。

1. 开环直流调速控制系统组成

开环控制系统是根据给定的控制量进行控制，而被控制量在整个控制过程中对控制量不产生任何影响。对于被控制量相对于其预期值可能出现的偏差，开环控制系统不具备修正能力。而直流调速开环控制系统通常是采用调节电枢电压方案，具体实现在 20 世纪 60 年代晶闸管整流器的应用而采用由晶闸管整流器和电动机（V-M）系统实现开环或闭环控制调速系统。由晶闸管整流器和电动机组实现开环系统结构。其中，晶闸管整流器提供可以调节直流电动机电枢电压实现直流电动机转速输出，而系统的输出量没有反馈给定环节参与控制实现转速的开环控制。



2. 开环直流调速控制系统仿真

（1）基于数学模型的开环直流调速系统仿真

1）开环直流调速控制系统数学模型。开环直流调速控制系统主要包括给定信号、晶闸管触发装置及整流环节、平波电抗器和直流电动机等 4 个主要环节。这里所说的基于数学模型的系统仿真主要是指基于传递函数的 MATLAB 下的 Simulink 下的实现，再通过机理法可以建立开环直流调速控制系统动态图。

然后，根据系统 I 直接给出各个环节的传递函数及参数，可以得到系统 I 开环控制的动态结构图。

2）开环直流调速系统仿真实现。根据系统 I 的开环系统动态结构图及其参数值，在 MATLAB 下的 Simulink 环境可以轻松地建立系统的仿真结构，如图 6-40 所示。电动机的转速输出动态曲线，如图 6-41 所示。

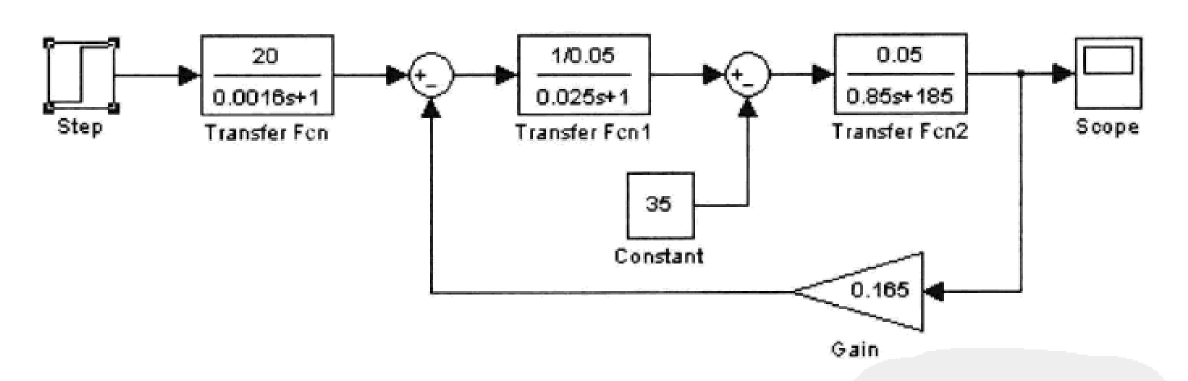


图 6-40 系统 I 仿真模型

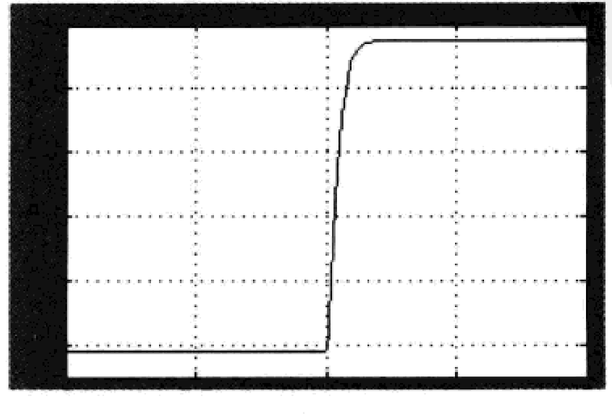


图 6-41 电动机转速输出曲线

通过改变给定信号的大小，来实现对电动机输出转速的控制与调节的目的。在仿真系统中的实现过程就是改变系统给定的阶跃信号的大小。但是开环控制系统的最大缺点是：无法实现的电网电压波动，以及电动机负载变化等扰动信号对转速影响。为了说明此问题，在上面的仿真系统在 4s 时刻将负载叠加一个阶跃信号，实现电动机的负载变化，最终的转速输出曲线如图 6-42 所示。可以看出由于负载的增加，使电动机的转速降低，而在系统稳定之后转速并没有恢复到原系统输出值。

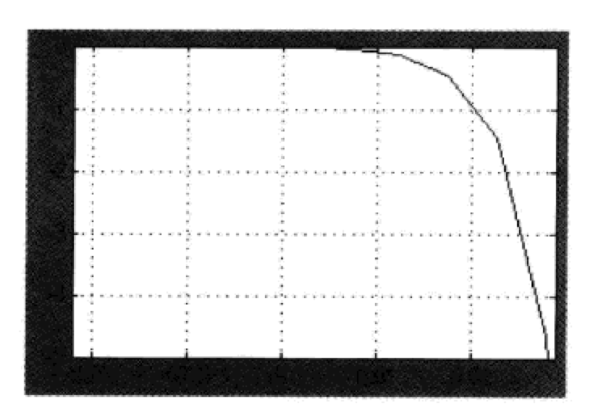


图 6-42 负载扰动情况转速输出曲线

（2）基于电气原理图的系统仿真

基于电气原理图实现系统的仿真实现主要过程是依据系统原理图，根据具体实现的不同功能，将整个系统划分成若干子模块，通过 MATLAB 中的 SimPowerSystems 工具箱电气元器件以及其他工具箱中的模块，组合实现子模块的仿真与建模，最终再依据电气原理图的电气连接实现各个子模块的连接即实现了整个系统的建模。开环直流调速控制系统原理图，如图 6-31 所示。下面以此为例讲解如何实现基于电气原理图实现系统的仿真过程。

1）三相对称交流电源模型。从 SimPowerSystems 工具箱中 Electrical Sources（电源）库中选择 AC Voltage Source（交流电压源）模块，参数设置如图 6-43 所示，即相电压峰值 220V，频率 50Hz，初始相位 0°，并将模块标签改为 A，表示为三相对称交流电源 A 相。通过复制得到两个电压源模块，更改两者参数初始相位为 120° 和 240°，即为 B、C 相电源，同样将模块标签改为 B、C。在 connectors 库中选择 Bus bar 和输出型接地模块，进行相应连接得到三相对称交流电源。

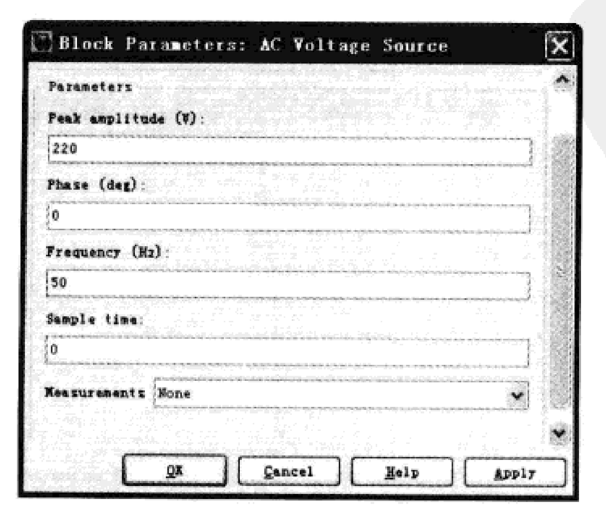


图 6-43 A 相电压源设置界面

2）晶闸管整流器模型。从 SimPowerSystems 工具箱中的 Power Electronics 库选取 Universal Bridge 模块，并将模块标签改为 SCR，然后双击模块图标，打开“Block Parameters: Universal Bridge”对话框进行参数设置，如图 6-44 所示。此参数设置一般要参考实际选取变流装置。



3）直流电动机模型。直流电动机模型选取在先前已经介绍过，参数设置如图 6-45 所示。

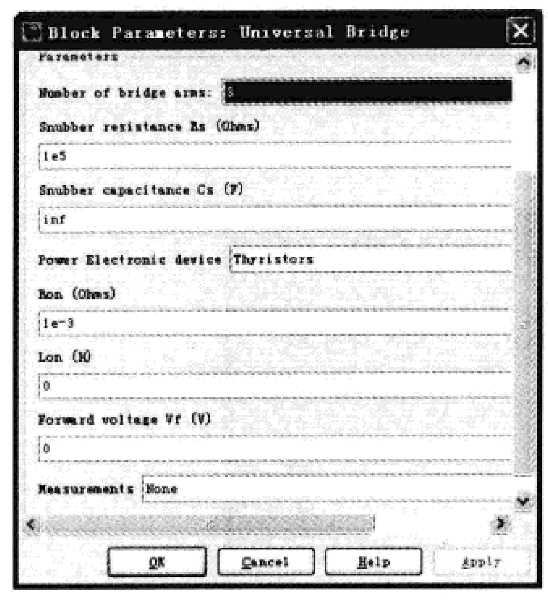



图 6-44　晶闸管整流器模型参数设置界面

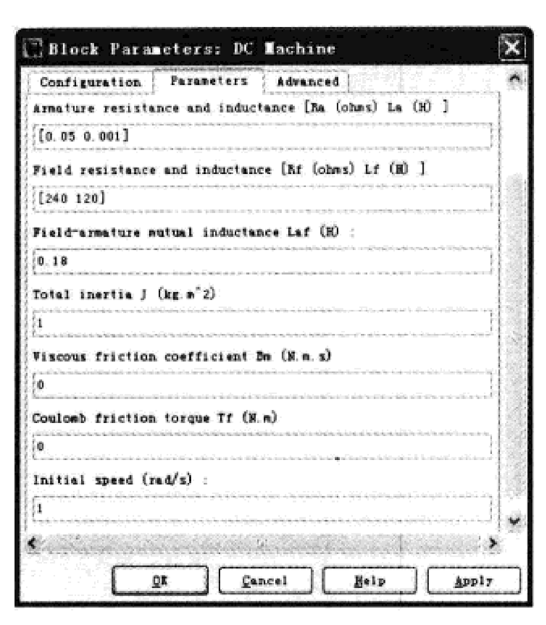


图 6-45　直流电动机参数设置

4）主回路平波电抗器模型。为了负载电流连续等原因，实际应用中在主电路中需要加入平波电抗器，建模过程是从 Elements 模型库中，选取 series RLC branch 模型。在 MATLAB 工具箱中没有纯电阻、纯电容和纯电抗器，而 Elecments 模型库有 series RLC branch 和 parallel RLC branch 元件，通过对两者的参数进行设置，可以实现纯电阻、纯电容和纯电气元器件。各参数设置分别见表 6-1 和表 6-2。

表 6-1　串联支路参数

名　称	参　数		
	Resistance R (ohms)	Inductance L/H	Capacitance C/F
纯电阻	R	0	Inf
纯电容	0	0	C
纯电抗	0	L	Inf

表 6-2　串联支路参数

名　称	参　数		
	Resistance R (ohms)	Inductance L/H	Capacitance C/F
纯电阻	R	Inf	0
纯电容	Inf	Inf	C
纯电抗	Inf	L	0

5）同步脉冲触发器模型。为了使得仿真模型简洁，对各个功能模块进行子系统封装，但是此过程在要封装的子系统与其他模块电气连接之后进行的。同时为了方便电气连接可以更改子系统的端口号，MATLAB 会自动的按照端口按照端口号大小顺序从上到下进行排列。将六脉冲同步触发器模型进行了子系统封装，并将子系统标签改为脉冲触发器。这里取得是线电压同步信号，因此在阻性负载时脉冲触发移相范围是 50°～180°，而且 50° 对应的是最大整流输出，180° 对应最小整流输出。

6）其他模块。构造开环直流电动机控制系统，需要使用的模块还有：L connector（L 型连接器）、3 个 Constant（常数模块）以及用来观测电动机变量的 4 个 To Workspace（输出到工作空间模块），并设置输出变量名分别为 w、I_a、I_f、T_e，w 参数设置，如图 6-46 所示。采用此方式输出可以看到系统变量主要变化过程。

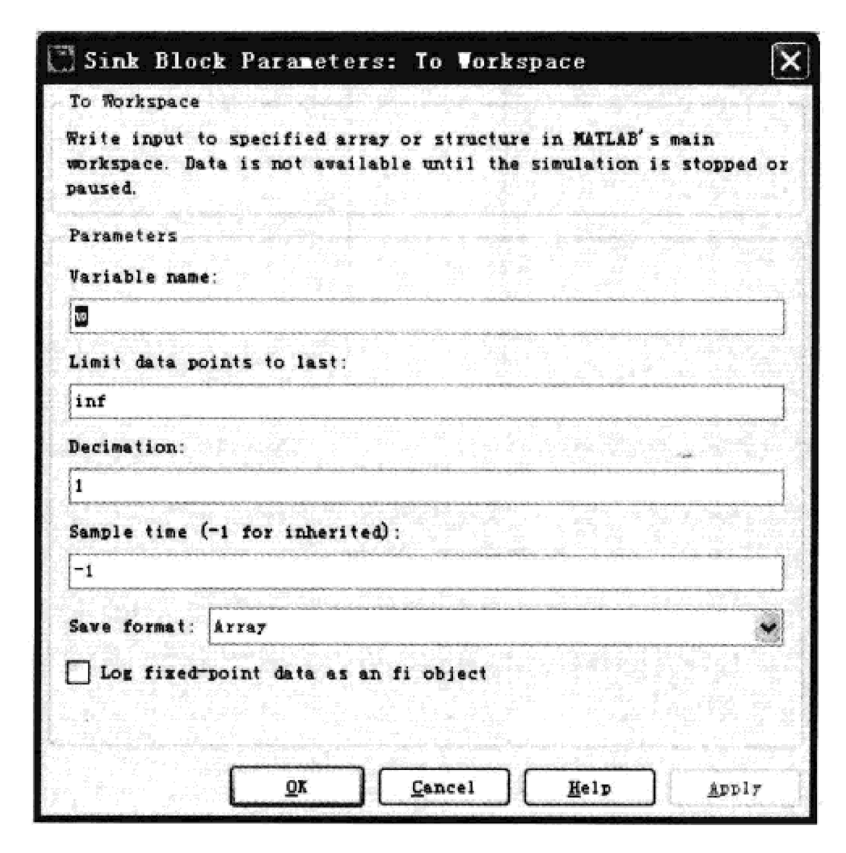


图 6-46 w 参数设置界面

将各个功能单元依据电气原理图，进行相应电气连接，最后得到开环直流电动机控制系统的仿真模型，如图 6-47 所示。

在完成了系统仿真模型的建立之后，便是对系统调试过程，同时也是对系统进行参数修改完善过程。首先在 MATLAB 的模型窗口打开 Simulation（仿真）菜单，进行 Configuration Parameters（仿真参数）设置，其参数设置如下：

- Start time（开始时间）：0。
- Stop time（停止时间）：10。
- Type（求解程序类型）：Variable-step（可变步长），ode23s（stiff/MOD Rosenbrock）。
- Max step size（最大步长）：auto（自动）。
- Min step size（最小步长）：auto（自动）。

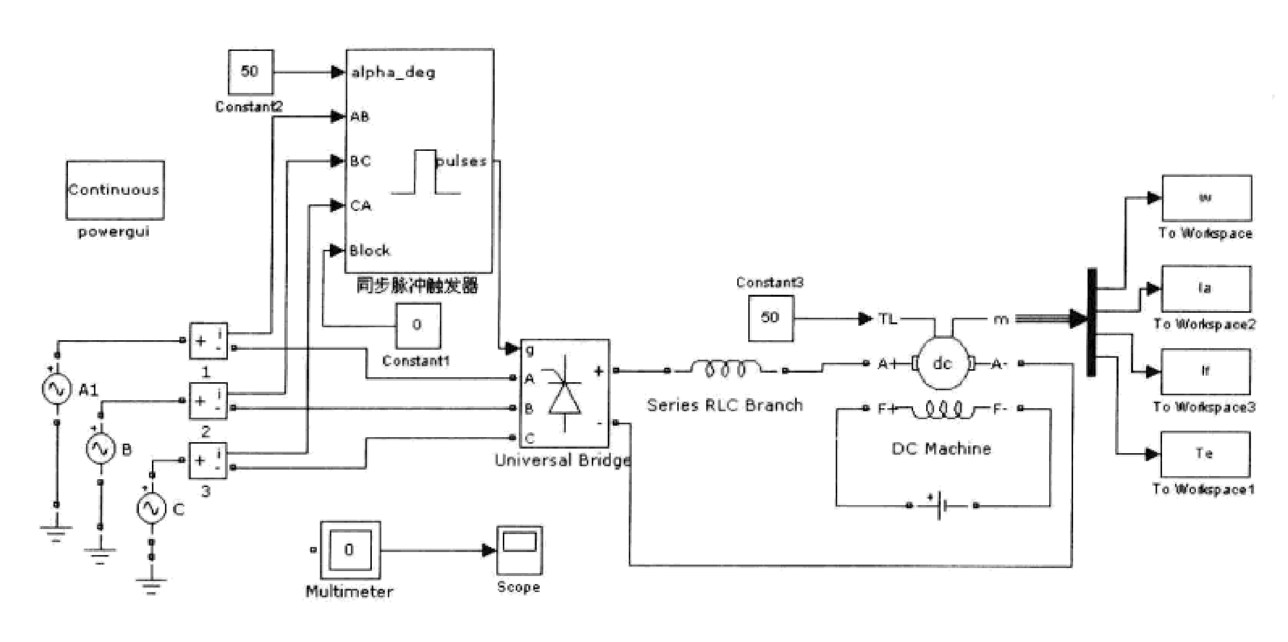




图 6-47 开环直流电动机控制系统仿真模型

- Initial step size（初始步长）：auto（自动）。
- Relative tolerance（相对容差）：1e-3。
- Absolute tolerance（绝对容差）：auto。

在 MATLAB 模型窗口中单击“Simulation”菜单下的“Start”选项，系统模型运行，到达设定的仿真时间而终止，并输出仿真结果。这里选取的是工作空间输出，以变量形式存在。

6.2.3 直流调速闭环控制系统仿真分析

1. 双闭环系统控制系统

（1）闭环直流调控制系统

闭环控制系统是既有参考输入控制输出量的前身或称顺向控制作用，又有输出量引回到输入端的反向控制作用，形成一个闭环控制形式。通常把输出量引回到输入端与参考输入量进行比较的过程称作反馈，所以闭环控制系统又称反馈控制系统。如果反馈信号与参考输入信号符号相反，称作负反馈；符号相同称作正反馈，自动控制系统中多采用负反馈。

在直流闭环控制系统中根据引入反馈信号的类型与结构形式的不同，在实际应用中看见遇到的系统有转速单闭环负反馈控制系统，电压负载控制系统，电压负反馈带电流补偿控制系统，以及双闭环控制系统，甚至多环控制系统。其中，最为常用的是转速单闭环负反馈控制系统和电流、转速双闭环直流调速控制系统，而转速单闭环负反馈控制系统包含在双闭环直流调速控制系统之中，因此下面主要讲解双闭环直流调速控制系统的仿真与建模。

（2）电流、转速双闭环直流调速控制系统

电流、转速双闭环直流调速控制系统由电流调节器和转速调节器串级联接而形成电流负反馈内环和转速负反馈外环构成。

转速调节器的输出作为电流调节器的输入，由电流调节器的输出去控制晶闸管整流器的触发器。通过设置转速调节器的输出限幅以及配合调节转速反馈通道的增益，可以得到电动机起动、制动等过程中的电枢回路的最大电流值，使得电动机快速起动与制动。同时通过双

环结构可以很好地抑制电网电压波动和负载变化等扰动量的电动机转速输出的影响，因此电流、转速双闭环直流调速控制系统具有良好的动态与静态特性。电流、转速双闭环直流调速控制系统设计除了对主电路与控制电路设计之外更为重要的环节就是电流调节器和转速调节器的设计一般电流与转速调节器比例积分（PI）调节器构成，因而便是两者比例积分参数整定的问题。参考双闭环的结构图和一些电力电子的知识，采用机理分析法可以得到双闭环系统的动态结构图，如图 6-48 所示。

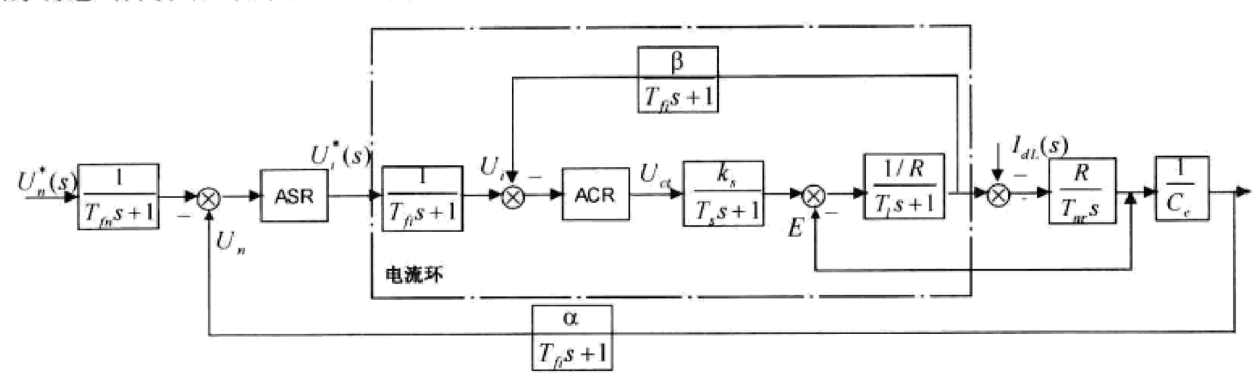


图 6-48　双闭环直流调速控制系统动态结构

2. 双闭环直流调速系统工程设计示例

双闭环直流调速系统工程设计步骤：首先根据工艺对电流的要求，设计内环-电流环，确定电流调节器的类型与参数，并确定电流调节器的组成电路与电路元器件；然后将电流环等效成一个小惯性环节，作为外环-转速环的一部分，再根据工艺对转速的要求采用同样方法设计转速环和转速调节器。

已知直流调速系统 I，实际生成工艺要求如下。

- 系统无静差。
- 电流超调量为 $\sigma_i < 5\%$ 。
- 在额定负载下，起动至额定转速的超调量 $\sigma_{\text{nnom}} < 5\%$

（1）系统参数计算

1）固有参数包括如下内容：

- 电势常数：$C_e = \dfrac{U_{\text{nom}} - I_{\text{nom}}R_a}{n_{\text{nom}}} = \dfrac{220 - 700\times 0.05}{1000}\text{V}/(\text{r}\cdot\text{min}^{-1}) = 0.185\text{V}/(\text{r}\cdot\text{min}^{-1})$ 。
- 常数转矩常数：$C_m = \dfrac{C_e}{1.03} = \dfrac{0.185}{1.03}\text{kg}\cdot\text{mA} = 0.182\text{kg}\cdot\text{mA}$ 。
- 电磁时间常数：$T_d = \dfrac{L_d}{R_d} = \dfrac{2\times 10^{-3}}{0.08}\text{s} = 0.025\text{s}$ 。
- 机电时间常数：$T_m = \dfrac{GD^2 R_d}{375 C_m C_e} = \dfrac{125\times 0.08}{375\times 0.185\times 0.182}\text{s} = 0.8\text{s}$ 。
- 晶闸管整流装置滞后时间常数：$T_s = \dfrac{1}{2mf} = \dfrac{1}{2\times 6\times 50}\text{s} = 0.0017\text{s}$ 。

2）预置参数包括如下内容：

- 选取转速输出限幅值：$U_{km} = 10\text{V}$ 。
- 晶闸管装置放大系数：$K_s = \dfrac{U_{d0}}{U_{km}} = \dfrac{1.05\times U_{\text{nom}}}{U_{km}} = \dfrac{1.05\times 220}{10} = 23$ 。

- 启动电流：$I_{dm}=1.5I_{\text{nom}}=1.5\times700\,\text{A}=1050\,\text{A}$。
- 选取转速调节器输出限幅值：$U_{im}=10\text{V}$。
- 电流反馈系数：$\beta=\dfrac{U_{im}}{I_{dm}}=\dfrac{10}{1050}\,\text{V/A}=0.0095\,\text{V/A}$。
- 选取电流反馈滤波时间常数：$T_{fi}=0.002$。
- 选取转速最大给定值：$U_{nm}^{*}=10\,\text{V}$。
- 可以得到转速反馈系数：$T_{fi}=0.002\alpha=\dfrac{U_{nm}^{*}}{n_{\text{nom}}}=\dfrac{10}{1000}\,\text{V}/(\text{r}\cdot\text{min}^{-1})=0.01\text{V}/(\text{r}\cdot\text{min}^{-1})$。
- 再取转速反馈滤波时间常数：$T_{fi}=0.01\text{s}$。

根据双闭环直流调速控制系统动态结构图可以得到系统 I 的双闭环动态结构图，如图 6-49 所示。



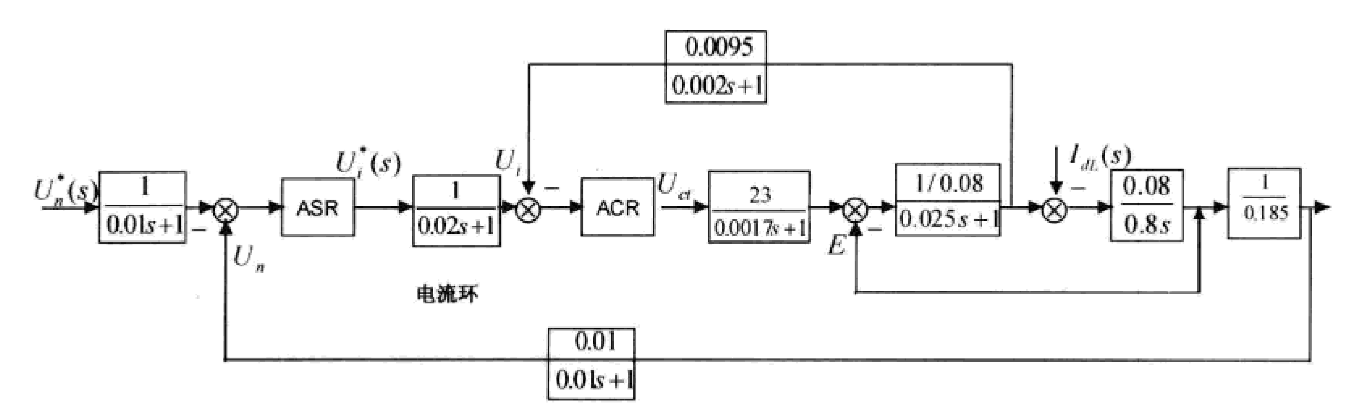


图 6-49 系统 I 动态结构

（2）系统设计

1）电流调节器设计。电流环主要作用是限制电流，因此一般将电流环校正为典型 I 系统，而电流调节器采用 PI 调节器。

$$W_{ACR}=K_i\frac{\tau_i s+1}{\tau_i s}$$

根据典型 I 系统设计可以得到如下结果：

$$\tau_i=T_l=0.025\text{s}$$
$$T_{\Sigma i}=T_s+T_{fi}=0.0017+0.002+0.0037\text{s}$$
$$K_i=\frac{T_l T_{\Sigma i}}{2\beta K_s T_{\Sigma i}}=\frac{0.025\times0.08}{2\times0.0095\times23\times0.0037}=1.24$$

2）转速调节器设计。转速的超调与动态速降均可由抗扰指标衡量，而抗扰指标以典型 II 系统为佳，因此转速调节器采用 PI 调节器，按典型 II 系统设计，取 $h=5$。

设转速调节器为：

$$W_{ASR}=K_n\frac{\tau_n s+1}{\tau_n s}$$

根据典型 II 系统设计可以得到如下结果：

$$T_{\Sigma n} = 2T_{\Sigma i} + T_{fn} = 2 \times 0.0037 + 0.01 = 0.0174\text{s}$$

$$\tau_n = 5T_{\Sigma n} = 5 \times 0.0174 = 0.087\text{s}$$

$$K_n = \frac{h+1}{2} \bullet \frac{\beta C_e T_m}{\alpha R_\Sigma T_{\Sigma n}} = \frac{6 \times 0.0095 \times 0.185 \times 0.8}{2 \times 5 \times 0.1 \times 0.08 \times 0.0174} = 60.6$$

3. 双闭环直流调速控制系统仿真

（1）基于数学模型的双闭环直流调速控制系统仿真

通过工程设计的方法建立的转速电流双闭环控制系统并确定了控制器的结构及其参数，也就是说，得到了双闭环的数学模型，因此可以实现基于数学模型的双闭环直流调速控制系统仿真。参照开环系统数学模型的仿真方法很容易建立双闭环系统的 Simulink 的实现，系统模型如图 6-50 所示。

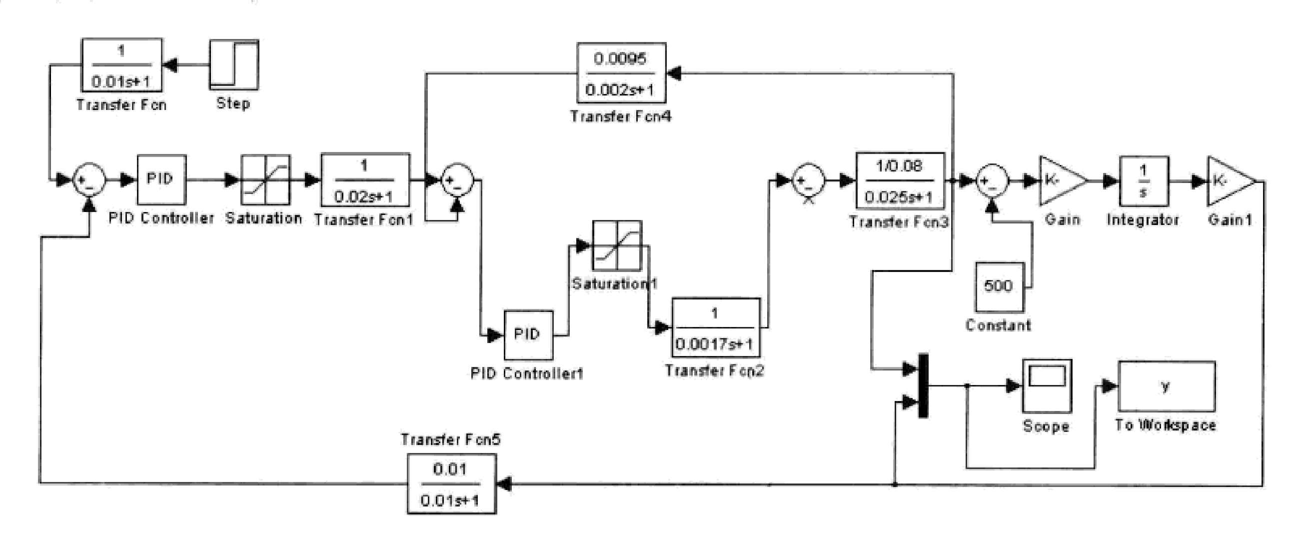


图 6-50　转速电流双闭环系统模型

仿真参数选择 ode23；开始时间 Start time 设为 0，停止时间 Stop time 设为 10，其他设置可以参考开环系统仿真设置。

完成建模和参数设置后，可以开始仿真运行。电枢电流和电动机转速输出曲线，如图 6-51 所示。可以比较清楚地看到双闭环在起动过程的强迫建流、恒流升速、速度调节等主要几个阶段。

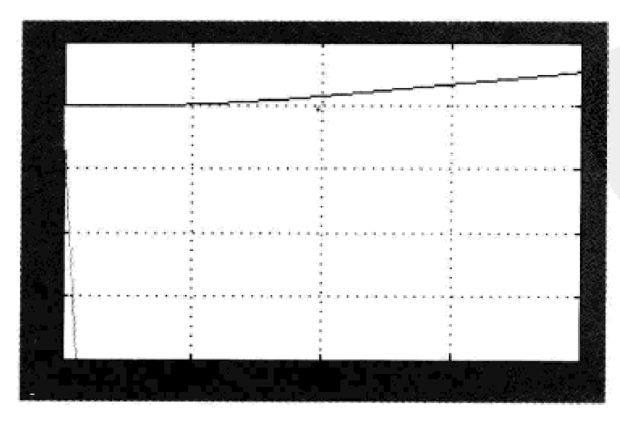


图 6-51　电枢电流和电动机转速输出曲线

（2）基于电气原理图的双闭环直流调速控制系统仿真

根据电流、转速双闭环直流调速控制系统原理图，在 MATLAB 模型窗口下建立双闭环

控制系统的仿真模型，如图 6-52 所示。为了能够比较建模两种方式的仿真效果，模型主电路的建立可以参照基于前面仿真模型创建过程。控制电路与参照基于数学模型的双闭环控制电路一样。同时参照系统 I 确定电机模型参数设置，如图 6-45 所示，晶闸管整流器的参数设置，如图 6-53 所示。平波电抗器参数选择为 0.001（H）。为了使得电流调节器（ACR）输出值与 6 脉冲同步触发器信号相应，在电流调节器输出加入[130 50]的非线性限幅模块，同时加-180V 的偏置电压。



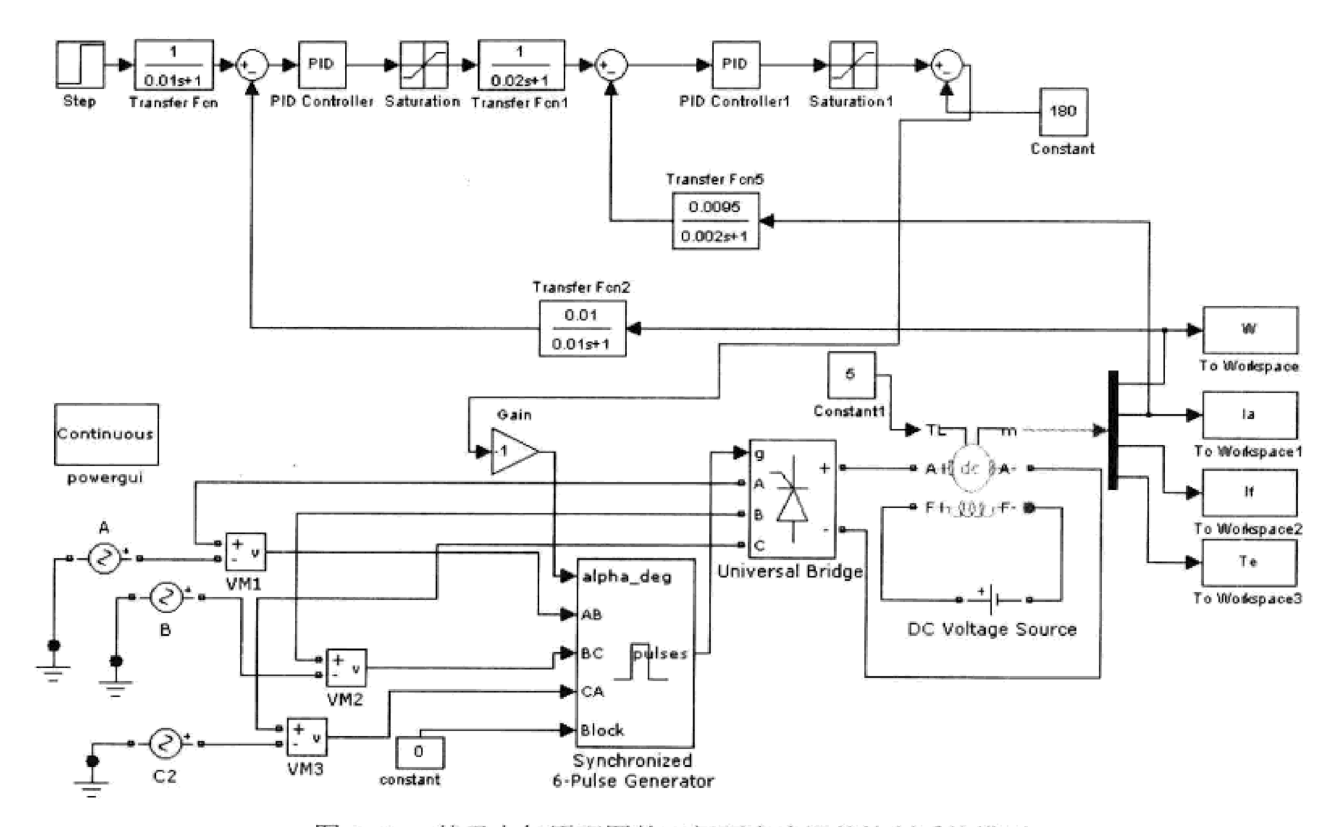


图 6-52　基于电气原理图的双闭环直流调整控制系统模型

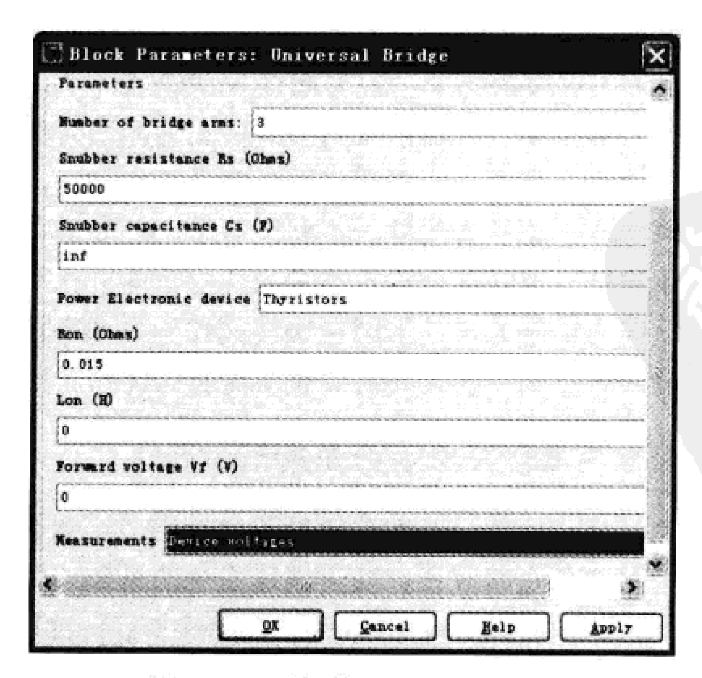


图 6-53　晶闸管整流器参数设置

主要的仿真参数设定包括：转速调节器比例系数常数为 60，积分系数 11.5；电流调节器比例系数 1.24，积分系数为 40。仿真算法选择 ode15s。

4. 仿真结果分析

从两种方法对转速电流双闭环系统进行了建模与仿真，分析系统输出，得到如下结论：

1）利用转速调节器的饱和特性，使系统保持恒定最大允许电流，在尽可能短的时间内建立转速，在退饱和和实现速度的调节和实现系统的无静差特性。

2）由于构成了无静差系统，在负载变化和电网电压波动等扰动的情况下，保持系统的恒定输出。

3）转速电流双闭环系统可以很好的克服负载变化和电网电压波动等扰动影响，特别是电网电压扰动点在电流环内，多数情况可以在电流环内就克服，而不会造成电动机转速的波动。

基于数学模型的双闭环系统与基于电气原理图的双闭环系统两种仿真方法得到相近的结果，同时说明仿真结果的正确性。不同之处在于两者仿真工作量的侧重点不同，基于数学模型的双闭环系统模型仿真方法主要工作量在系统的数学模型的建立和控制器的设计方面，而基于电气原理图的双闭环系统仿真方法主要工作量在模型参数设置和控制器的设计，以及系统的调试方面。

6.3 交流电动机系统建模与仿真分析

6.3.1 交流电动机调速原理

从电动机学可知，异步电动机的转速表达式为：

$$n = \frac{60 f_1}{n_p}(1-s) \tag{6-4}$$

式中，f_1 为电动机的定子供电频率；n_p 为电动机极对数；s 为转差率。

因此实现异步电动机输出速度的改变，主要通过 3 类方式来实现，即改变电动机的极对数、变化转差率以及改变供电频率。目前常见到的具体实现调速方案有：变极调速、调压调速、串级调速以及变频调速等。其中，变极调速方式属于有级调速，调速范围窄，应用场合有限；调压调速方式是消耗转差功率为代价，不利于节能，一般应用在中小型风机、泵类等功率调速系统中；串级调速是以消耗部分转差功率为代价较前者在节能方面略胜一筹，是一种结构简单，实现方便，较为经济方式，多用在绕线式异步电动电机技术改造中；变频调速是最为理想的异步电动机调速方式，以其高效率和高性能等优势，目前应用最为广泛。

6.3.2 Simulink 建模与仿真在交流调速系统的分析

交流电动机和直流电动机相比较，其数学模型要复杂得多，对交流电动机的建模与仿真更为复杂，而交流电动机的建模是研究设计交流调速系统的基础。为了简化交流电动机的建模复杂工作，MATLAB 推出的 SimPowerSystems 工具箱中定制封装了系列电动机模型，当然包括在先前介绍的直流电动机模型。SimPowerSystems 工具箱中的交流电动机模型位于 Machines 库，主要包括 Asynchronous Machine Pu Units（单位制的异步电动机）、

Asynchronous Machine SI Units（国际单位制的异步电动机）、Permanent Magnet Synchronous Machine（永磁式同步电动机）、Simplified Synchronous Machine Pu Units（单位制的简化同步电动机）、Simplified Synchronous Machine SI Units（国际单位制的简化同步电动机）等。本章采用国际单位制的异步电动机，下面以国际单位制的异步电动机为例介绍。

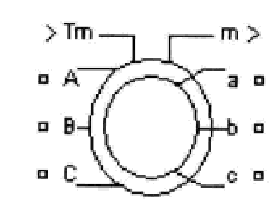


图 6-54 异步电机模型

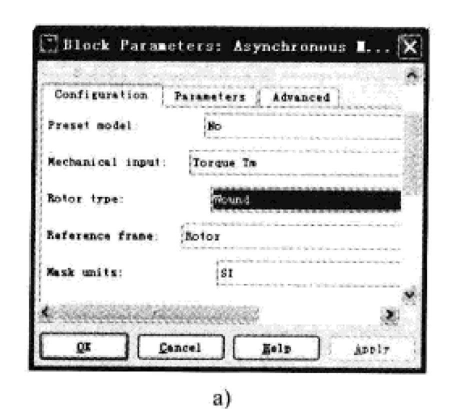


国际单位制的异步电动机模型有 4 个输入端子和 4 个输出端子，如图 6-54 所示。

其电气连接和功能如下：

- A，B，C：交流电动机的定子电压输入端子。
- Tm：电动机负载输入端子，一般是加到电动机轴上的机械负载。
- a，b，c：绕线式转子输出电压端子一般短接，而在笼式电动机为此输出端子。
- m：电动机信号输出端子，一般接电动机测试信号分配器观测电动机内部信号，或引出反馈信号。

国际单位制的异步电动机模型的参数设置，在确认异步电动机模型已经位于模型窗口中，双击电动机模块，打开“Block Parameters：Asynchronous Machine SI Units”对话框，如图 6-55a、b、c 所示。



a)



b)

c)

图 6-55 异步电动机模型参数设置

a) Configuration 下拉列表框 b) Advanced·下拉列表框 c) Parameters 下拉列表框

部分相关参数如下：

- Rotor type：转子类型列表框，分别可以将电动机设置为 Wound（绕线式）和 Squirrel-cage（鼠笼式）两种类型。
- Reference frame：参考坐标列表框，可以选择 Rotor（转子坐标系）、Stationary（静止坐标系）、Synchronous（同步旋转坐标系）。
- Nom.power，L-L volt.and freq.[Pn(VA), Vn(Vrms), fn(Hz)]：额定功率（V·A），线电压（V），频率（Hz）。
- Stator[Rs (ohm) Lls(H)]：定子电阻 Rs(ohm)和漏感 Lls（H）。
- Rotor [Rs' (ohm) Lls'(H)]：转子电阻 Rs(ohm)和漏感 Lls（H）。
- Mutual inductance Lm(H)：互感 Lm(H)。
- Intia，friction factor and pairs of poles [J (kg.m^2)]：转子惯量[J (kg.m^2)]，摩擦系数和极对数。
- Initial conditions [s() th(deg) isa isb isc (A)]：初始条件包括初始转差 s，点角度 phas，phbs，phcs(deg)和定子电流 i_{sa} i_{sb} i_{sc} (A)。

交流电动机模型输出不能直接得到，在仿真过程一般要和 Machines Measurement Demux（电机测试信号分配器）配合使用，下面进一步介绍电动机测试信号分配器功能和使用。电动机测试信号分配器模块位于 Machines 库之中，如图 6-56 所示。

电动机测试信号分配器有一个输入端子 m，此端子要和电机输出端子 m 相连接，输出端子最多由 21 路输出信号构成，而具体产生那些信号，需要在“Block Parameters：Machines Measurement Demux”对话框中选择相应的复选项，如图 6-57 所示。首先要选择电动机的类型，包括 Synchronous（同步电动机），Simplified Synchronous（简化同步电动机），Asynchronous（异步电动机），Permanent Magnet Synchronous（永磁同步电动机）等，当然选择不同的电动机将会有不同的输出信号设置，这里仍然以异步电动机为例。

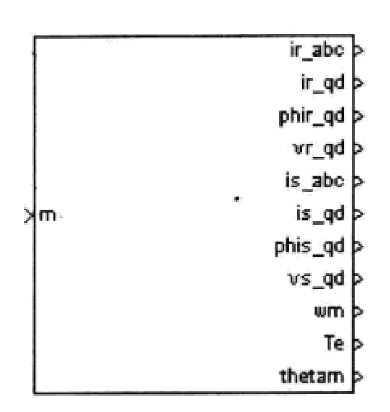


图 6-56 Machines Measurement Demux

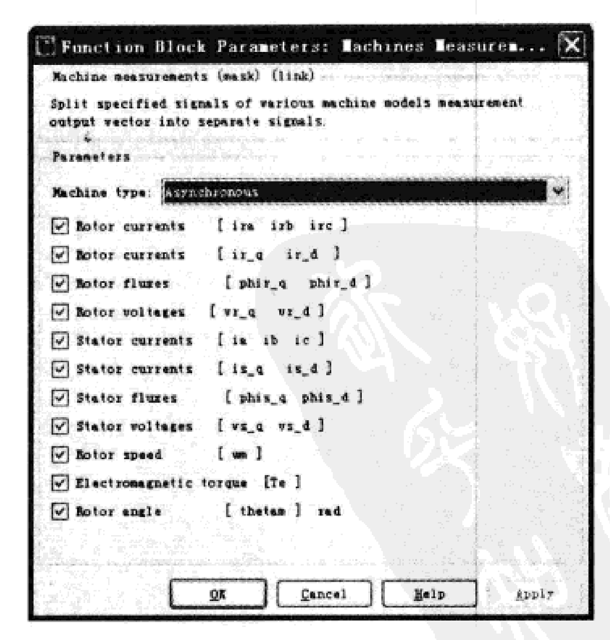


图 6-57 电动机测试信号分配器的输出设置

电动机测试信号分配器的输出信号构成如下。



- ira、irb、irc：转子电流。
- ir_q、ir_d：同步 d-p 坐标下的 q 轴下的转子电流和 d 轴下的转子电流。
- phir_qd：同步 d-q 坐标下的 q 轴下的转子磁通和 d 轴下的转子磁通。
- vr_q、vr_d：同步 d-p 坐标下的 q 轴下的转子电压和 d 轴下的转子电压。
- is_a、is_b、is_c：定子电流。
- is_q、is_d：同步 d-q 坐标下的 q 轴下的定子电流和 d 轴下的定子电流。
- phis_q、phis_d：同步 d-q 坐标下的 q 轴下的定子磁通和 d 轴下的定子磁通。
- vs_q、vs_d：同步 d-q 坐标下的 q 轴下的定子电压和 d 轴下的定子电压。
- wm：电动机的转速。
- Te：电动机的机械转矩。
- thetam：电动机转子角位移。

上面只是为电动机建模设置参数提供参考，而在具体实例建模和仿真过程中，还要根据具体的应用选择的电动机类型以及电动机的参数来设置具体的参数，而且是一个反复修正调整过程。

6.4 电力系统时域分析

6.4.1 电力系统不对称运行分析法

电力系统正常运行时可以认为是三相对称的，即每个元器件三相阻抗相同，各处三相电压和电流对称，且具有正弦波形和正常相序。当电力系统发生不对称短路或个别地方一相或两相断开时，则对称运行方式遭到破坏，三相电压和电流将不对称，而且波形发生不同程度的畸变，即除基波形，还含有一系列谐波分量。一般情况下，在电力系统分析中，对于不对称故障采用简单的对称分量法进行分析。

对称分量法是指任意不对称的三相相量均可以分解为三组相序不同的对称分量：正序分量、负序分量和零序分量。它们之间的数学关系如下：

$$\begin{pmatrix}\dot{F}_{a1}\\ \dot{F}_{a2}\\ \dot{F}_{a0}\end{pmatrix}=\frac{1}{3}\begin{pmatrix}1 & e^{j120°} & e^{j240°}\\ 1 & e^{j240°} & e^{j120°}\\ 1 & 1 & 1\end{pmatrix}\begin{pmatrix}\dot{F}_{a}\\ \dot{F}_{a}\\ \dot{F}_{c}\end{pmatrix}$$

已知正序分量、负序分量和零序分量时，可以用下式合成三相相量。

$$\begin{pmatrix}\dot{F}_{a}\\ \dot{F}_{a}\\ \dot{F}_{c}\end{pmatrix}=\frac{1}{3}\begin{pmatrix}1 & 1 & 1\\ e^{j240°} & e^{j120°} & e^{j120°}\\ e^{j120°} & e^{j240°} & 1\end{pmatrix}\begin{pmatrix}\dot{F}_{a1}\\ \dot{F}_{a2}\\ \dot{F}_{a0}\end{pmatrix}$$

MATLAB 软件中的电力系统元器件库中提供了 3-Phase Sequence Analyzer（三相序分量分析）元器件，下面对其进行介绍。该元器件在电力系统元器件库的 Extras（附加）元器件库中的 Measurements（测量）元器件库中，其元器件模型如图 6-58 所示。

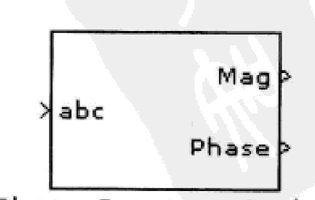


图 6-58 三相序分量分析元器件

双击三相序分量分析元器件，得到参数设置对话框，如图 6-59 所示。其中包括 3 个参数设置，分别为 Fundamental frequency f1（基频频率），用来设置三相输入信号的基频频率；Harmonic n（谐波次数），用来指定进行序分量分析的谐波；Sequence（序量选项）选项，用来选择显示的序分量，包括 4 个选项：Positive（正序分量）、Negative（负序分量）、Zero（零序分量）和 Positive Negative Zero（所有序分量）。

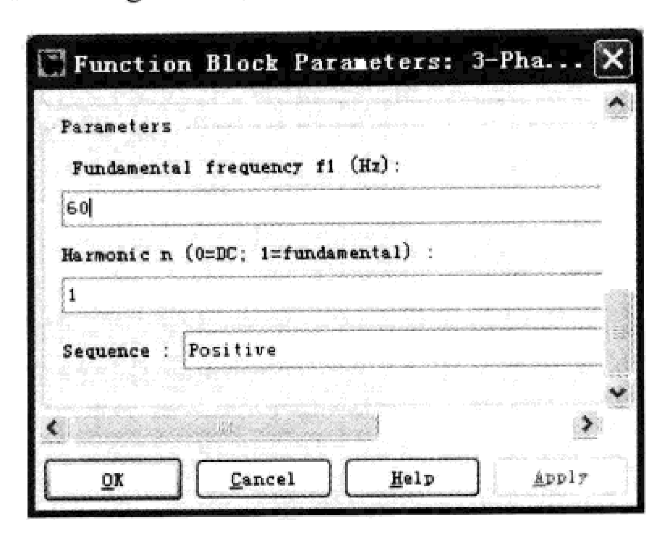


图 6-59　三相序分析元器件参数对话框

下面以一个简单的实例来说明三相序分量分析的方法。

【例 6-3】 三相序分量分析，设计给定的电路图模型，分析 A 相接地后，其正序、负序、零序分置的变化情况。

1. 电路图设计

按照前面介绍的建立电路图模型的方法建立三相序分量分析电路图模型，如图 6-60 所示。.

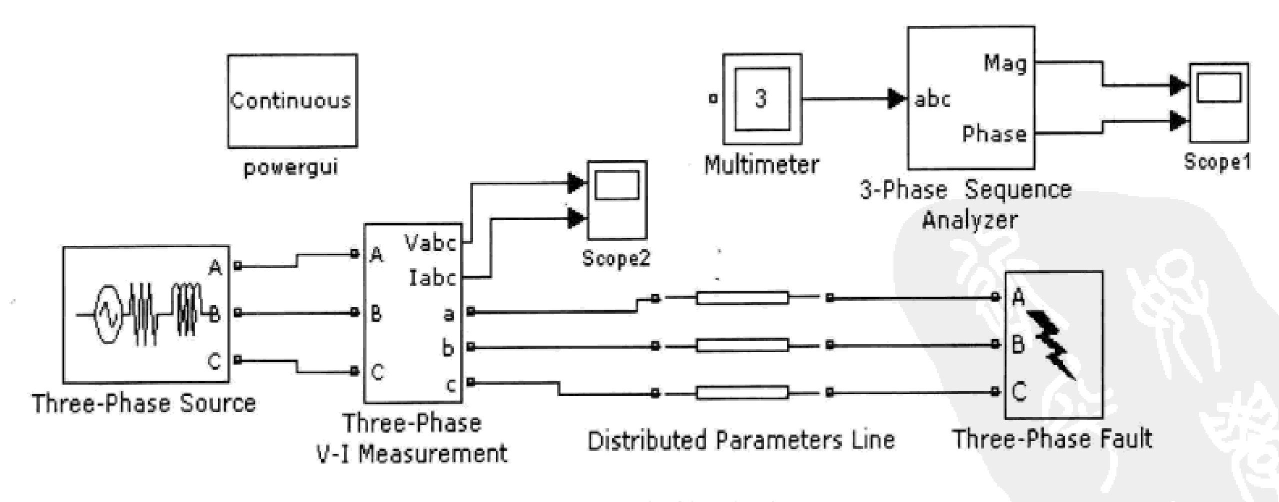


图 6-60　电路图模型

2. 模型参数设置

- 设置三相交流电压源的参数，如图 6-61 所示。
- 设置三相分布参数等值电路元器件，参数如图 6-62 所示。
- 设置三相短路元器件参数，如图 6-63 所示。

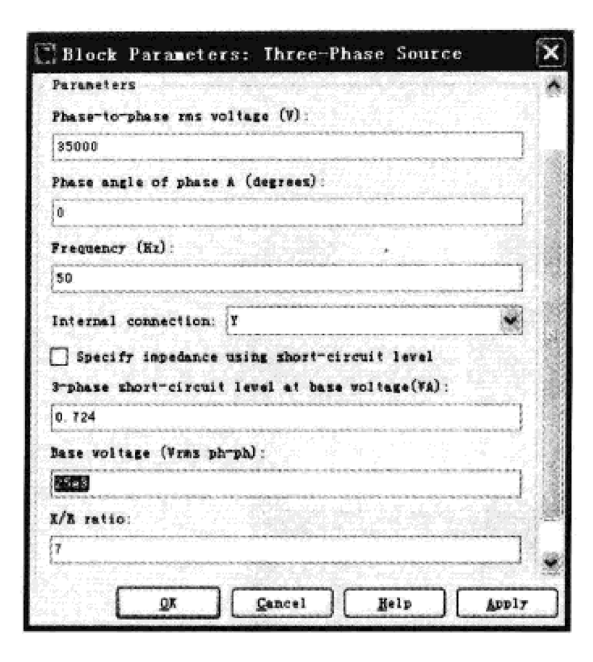


图 6-61　三相交流电压源的参数设置

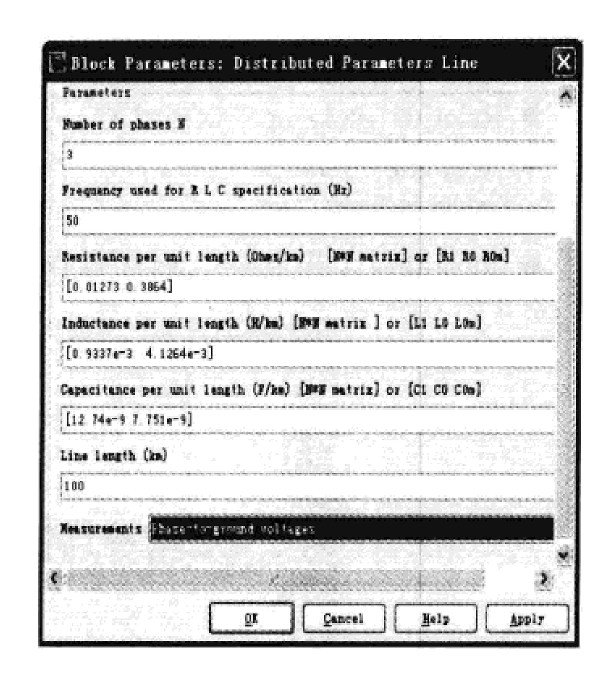


图 6-62　三相分布等值电路元器件的参数设置



● 设置三相序分量分析元器件参数，如图 6-64 所示。

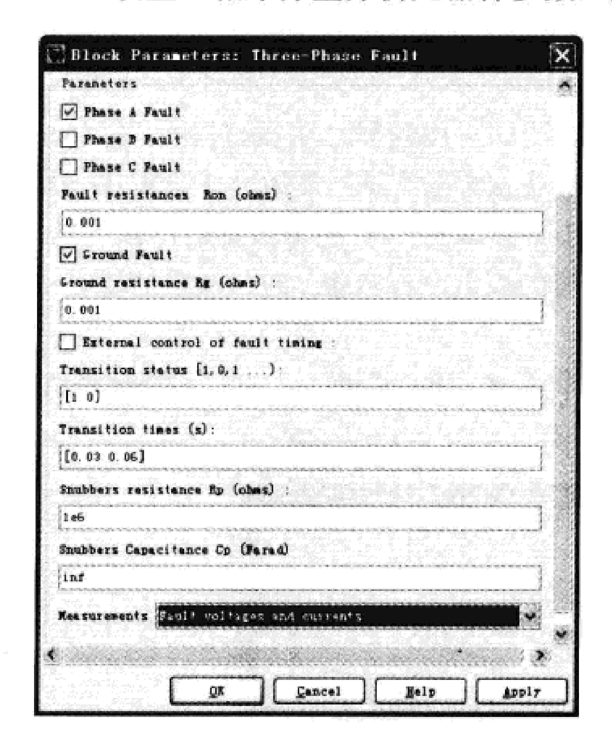


图 6-63　三相短路元器件的参数设置

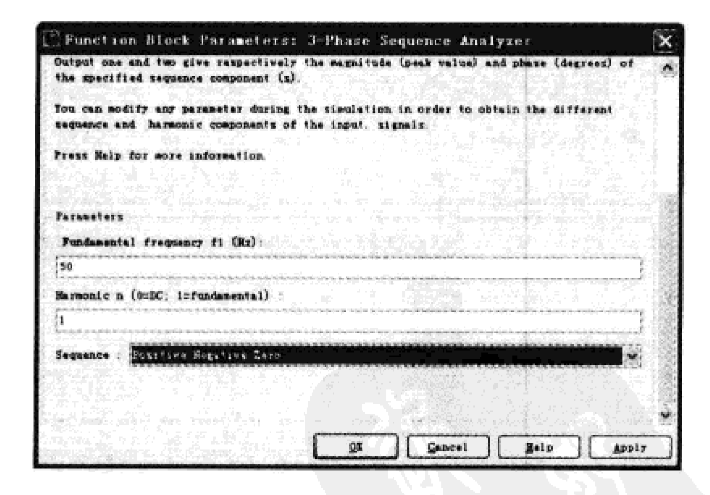


图 6-64　三相序分量分析元器件的参数设置

3. 仿真参数设置

对本系统仿真参数设置如下。

- Start time（开始时间）：0。
- Stop time（停止时间）：0.1。
- Type（求解程序类型）：Variable-step（可变步长），ode23tb（stiff/TR-BDF2）。
- Max step size（最大步长）：auto（自动）。
- Min step size（最小步长）：auto（自动）。

- Initial step size（初始步长）：auto（自动）。
- Relative tolerance（相对容差）：1e-3。
- Absolute tolerance（绝对容差）：auto。

4．仿真结果及分析

- 仿真得到 A 相单相接地时的三相电压和电流曲线，如图 6-65 所示。

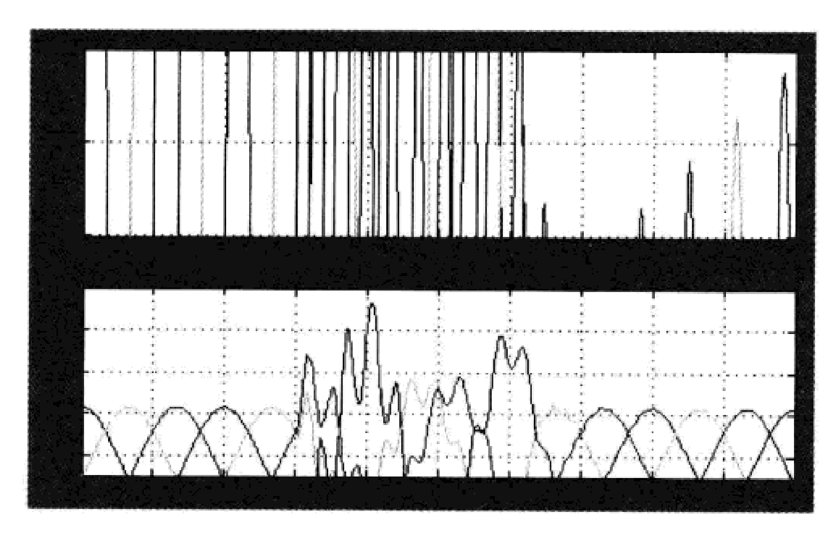


图 6-65　三相电压和电流曲线

在单相接地没有发生前，A、B、C 三相电压、电流均对称运行。在 0.03s 时，发生 A 相接地短路，此时三相电压、电流发生变化，A 相电压幅值迅速下降，其值大于零但小于相电压，B 相、C 相电压迅速上升，其值大于相电压但小于线电压；A 相电流幅值迅速上升，B 相、C 相电流也相对发生变化，但幅值小于 A 相电流的幅值。在 0.06s 时，故障解除，三相电压、电流又逐渐恢复为三相对称运行的状态。

- 在 Multimeter（万用表）元器件中选择故障点 A、B、C 电压。得到故障相 A 相电压的正序、负序、零序分量的幅值和相位，如图 6-66 所示。

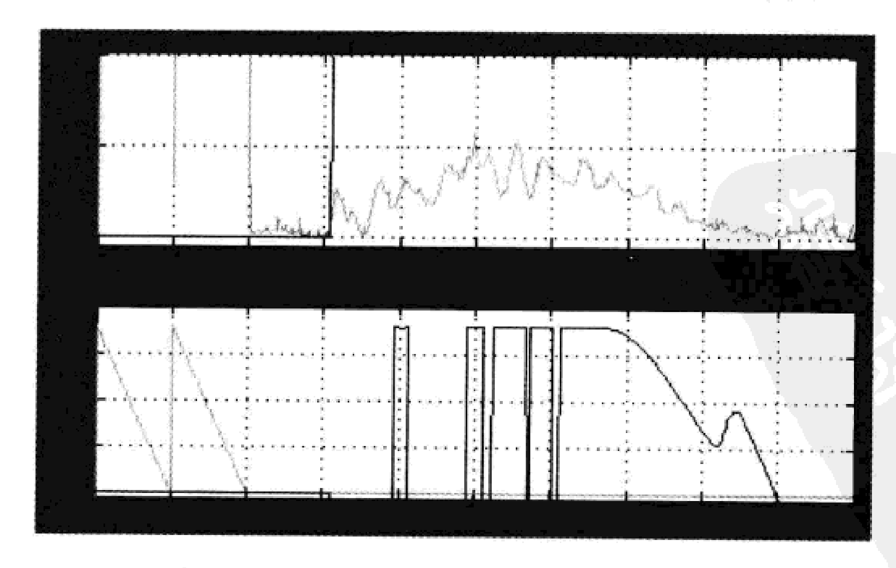


图 6-66　A 相电压的正序、负序、零序分量

- 在万用表元器件中选择故障点 A、B、C 电流。得到故障相 A 相电流的正序、负序、零序分量的幅值和相位，如图 6-67 所示。

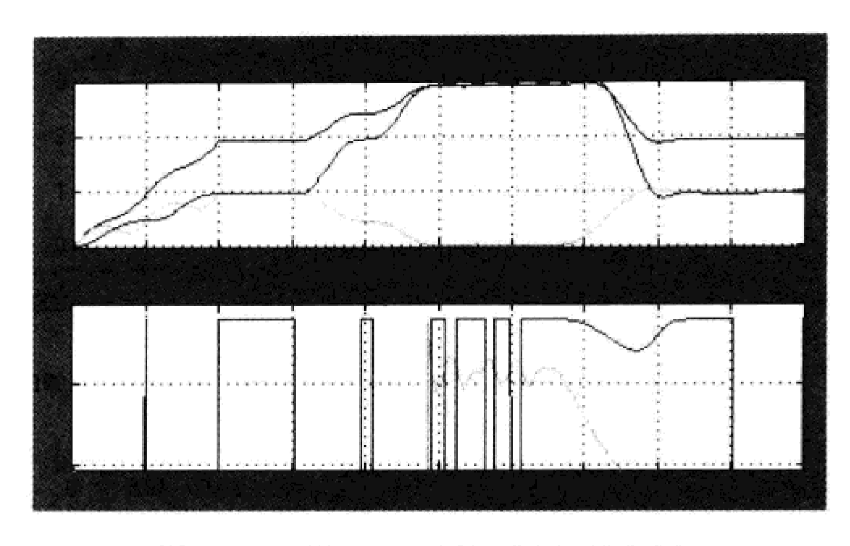


图 6-67　A 相电压的正序、负序、零序分量

在这种情况下，故障相 A 相电流的正序、负序、零序分量均相等。这和电力系统理论分析的结果相同，证明了此种仿真分析的方法是有效可靠的。

6.4.2 电力系统时域分析工具

MATLAB 软件提供了一个对电力系统和电路进行分析的 Powergui（用户界面）工具。元器件的模型如图 6-68 所示。



图 6-68　电力系统和电路进行分析的用户界面工具

双击元器件模型打开元器件的选项及参数对话框，如图 6-69 所示。

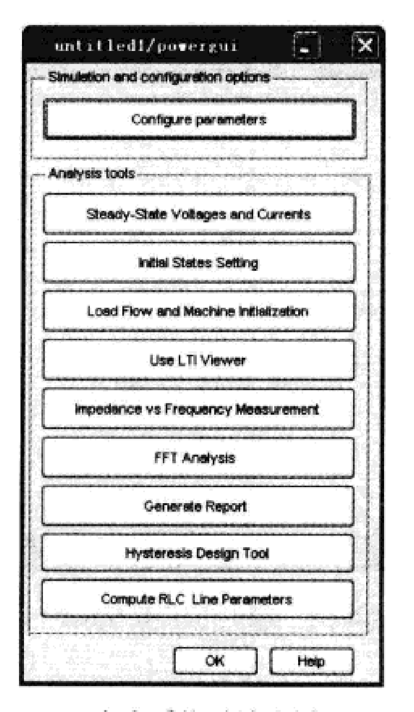


图 6-69　电力系统时域分析工具选项

【例 6-4】 电力系统时域分析工具简介。根据给出的电路图模型，利用电力系统时域分析工具来验证其功能。

其电路图模型如图 6-70 所示。

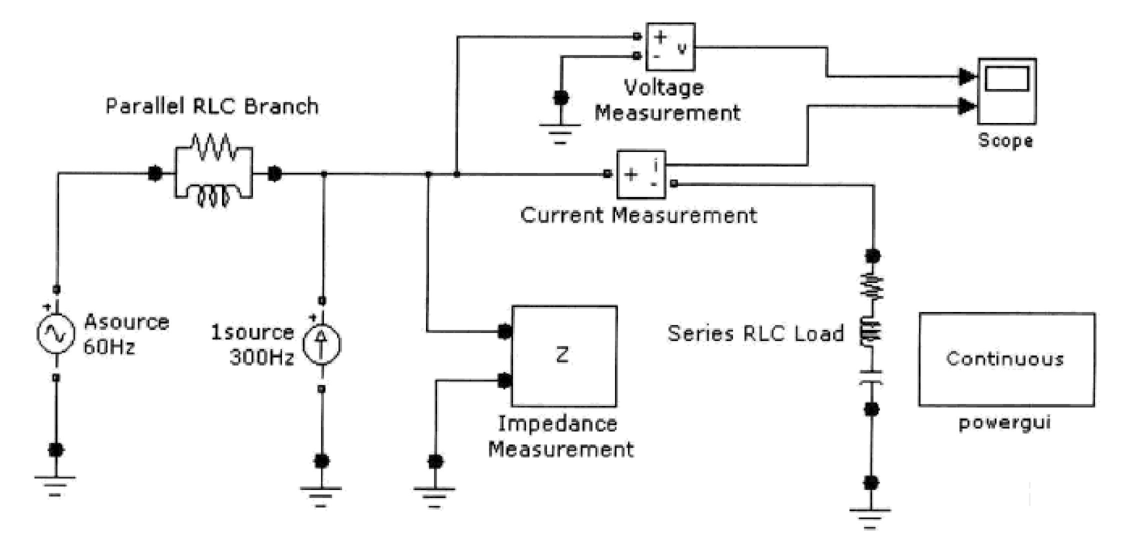


图 6-70 电路图模型

在进行仿真之后，双击电力系统 Powergui（时域分析工具）元器件，打开元器件的选项及输出对话框，选择“Steady-State Voltages and Currents”（稳态电压和电流）选项，则出现稳态电压和电流对话框，如图 6-71 所示。其中参数包括 Units（单位）选项，分为 Peak values（峰值）和 RMS value（有效值）两个选项；Frequency（频率）选项，包括了电力系统模型中的所有的电流频率，选择不同频率可以查看不同频率下稳态电压和电流；Display（显示）选项，包括 States（状态变量）复选框，Measurements（测量值）复选框，Sources（电源）复选框，Nonlinear elements（非线性量）复选框。选中相应的复选框，则显示相应的稳态电流和电压。

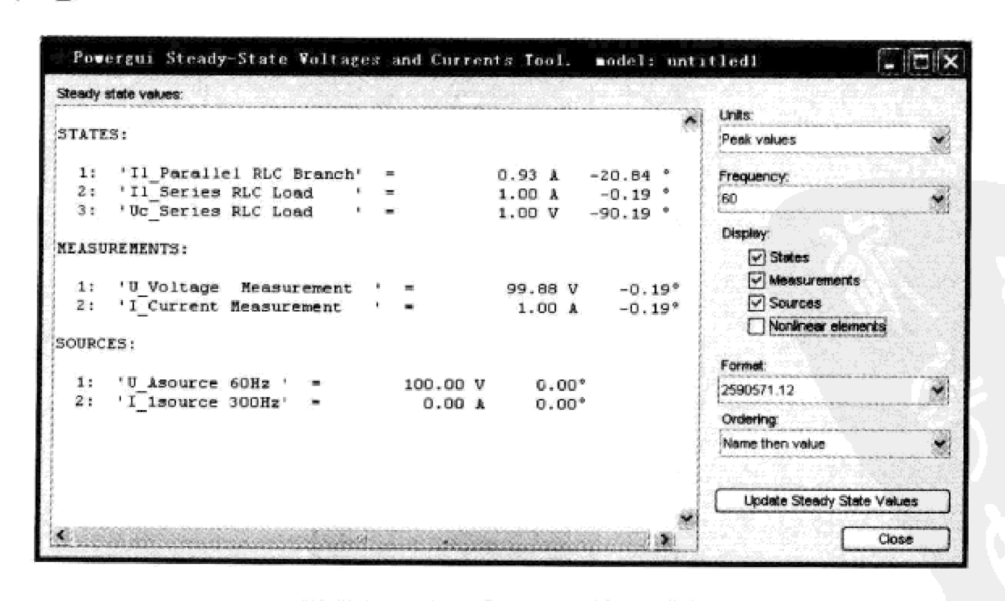


图 6-71 稳态电压和电流对话框

在此例中，显示的是频率为 60Hz 的所有变量的电压和电流峰值。

选择“Initial States Setting”（初始状态设置）选项，则打开初始状态设置对话框，如图 6-72

所示。

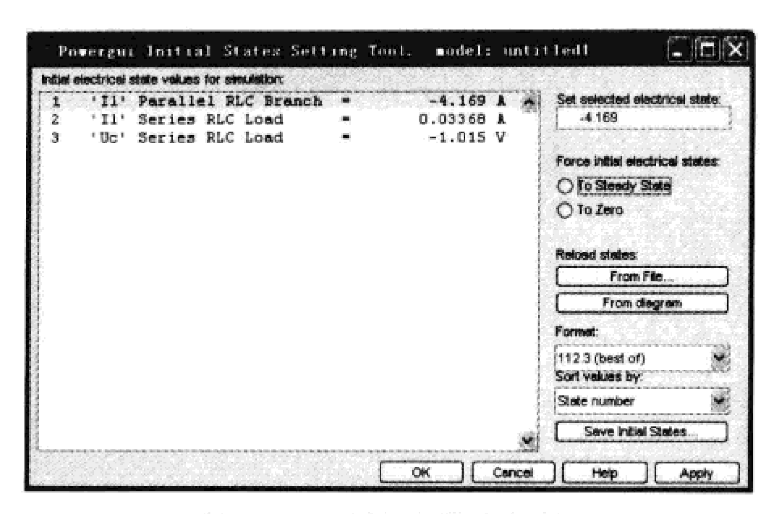


图 6-72　初始状态设置对话框

其中，初始的状态变量显示的是稳态的数值，可以通过“Set selected electrical state”（状态变量的选择设置）选项对其初始值进行设置。在此例中，将所有的状态变量初始值设置为0，则得到滤波器的输出为零状态响应，如图 6-73 所示。



图 6-73　零状态响应

选择“Use LTI View”（线性时变观察器）选项，则出现线性时变观察器对话框，如图 6-74 所示。其中，System inputs（系统输入量）在左侧显示，System outputs（系统输出量）在右侧显示。

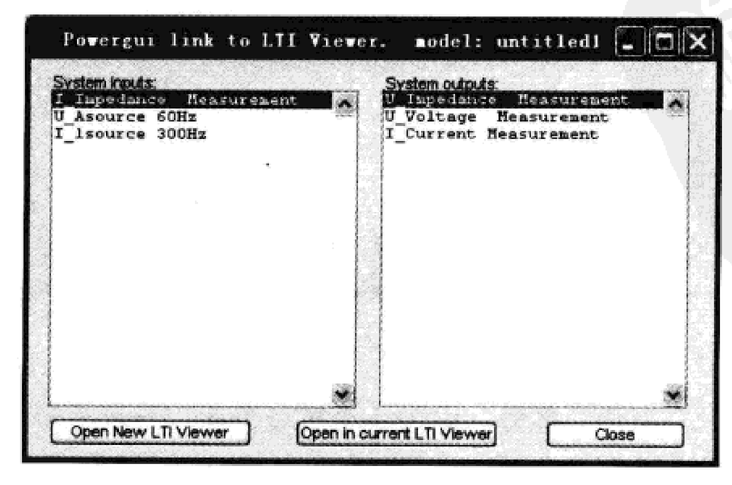


图 6-74　线性时变观测器对话框



在此例中选择阻抗测量元器件的电流作为系统输入量，选择阻抗测量元器件的电压作为系统输出量，得到阶跃响应曲线，如图 6-75 所示。利用这种方法还可以对脉冲响应曲线进行仿真绘制，因此该功能选项具有非常实用的应用价值。

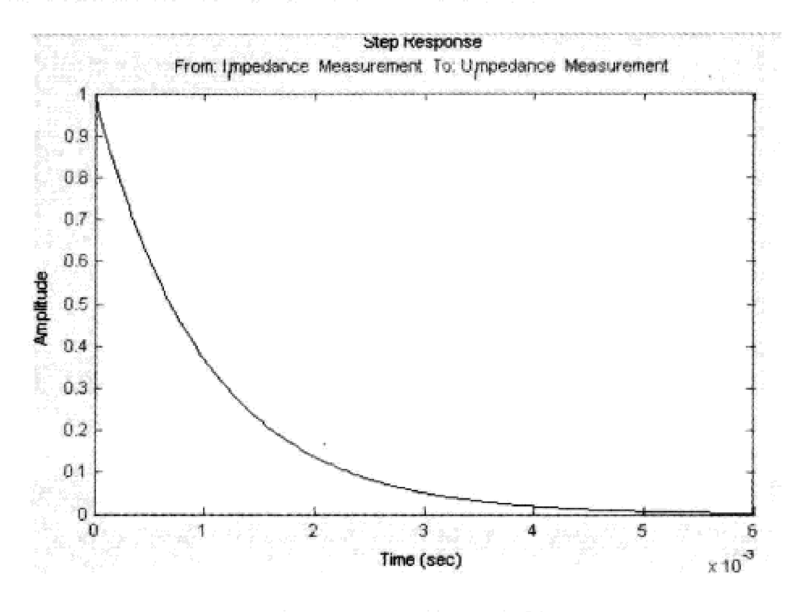


图 6-75　阶跃响应曲线

选择“Impedance vs Frequency Measurement”（阻抗或频率测量）选项，则出现阻抗或频率测量选项对话框，如图 6-76 所示。在选择了需要显示的频率响应的元器件后，则可以出现相应的变化曲线。在本例中，选择的是阻抗测量元器件，则出现阻抗值与频率之间的关系曲线。

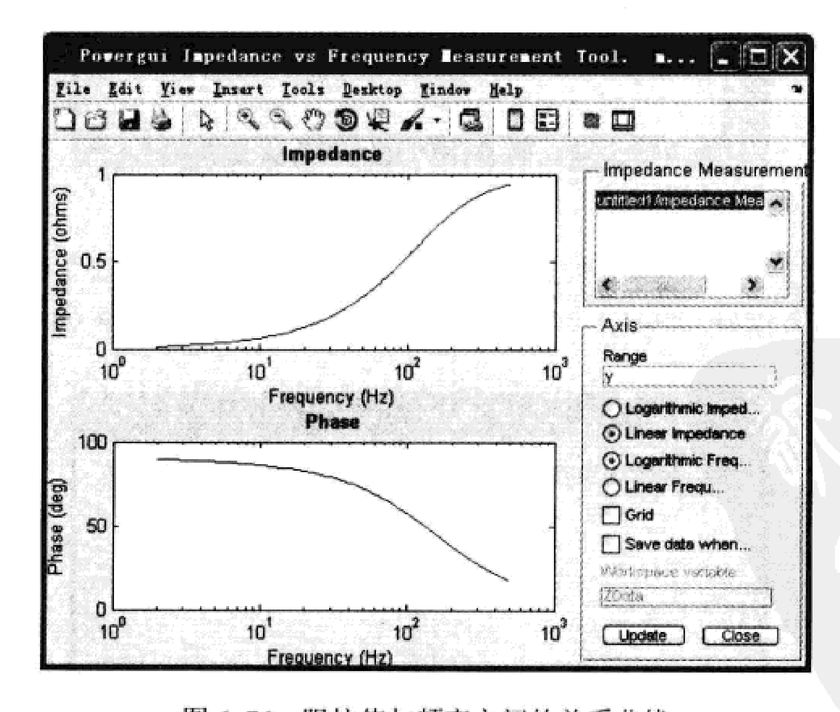


图 6-76　阻抗值与频率之间的关系曲线

除了上述已经进行介绍的功能外，电力系统时域分析工具还具有许多其他的功能。有兴趣的读者可以参考相关文献。

6.5 电力系统仿真示例分析

【例 6-5】 电力系统自动重合闸仿真分析，该系统电压等级为 220kV，为双电流供电系统。

（1）电路图设计

建立电路图模型，如图 6-77 所示。

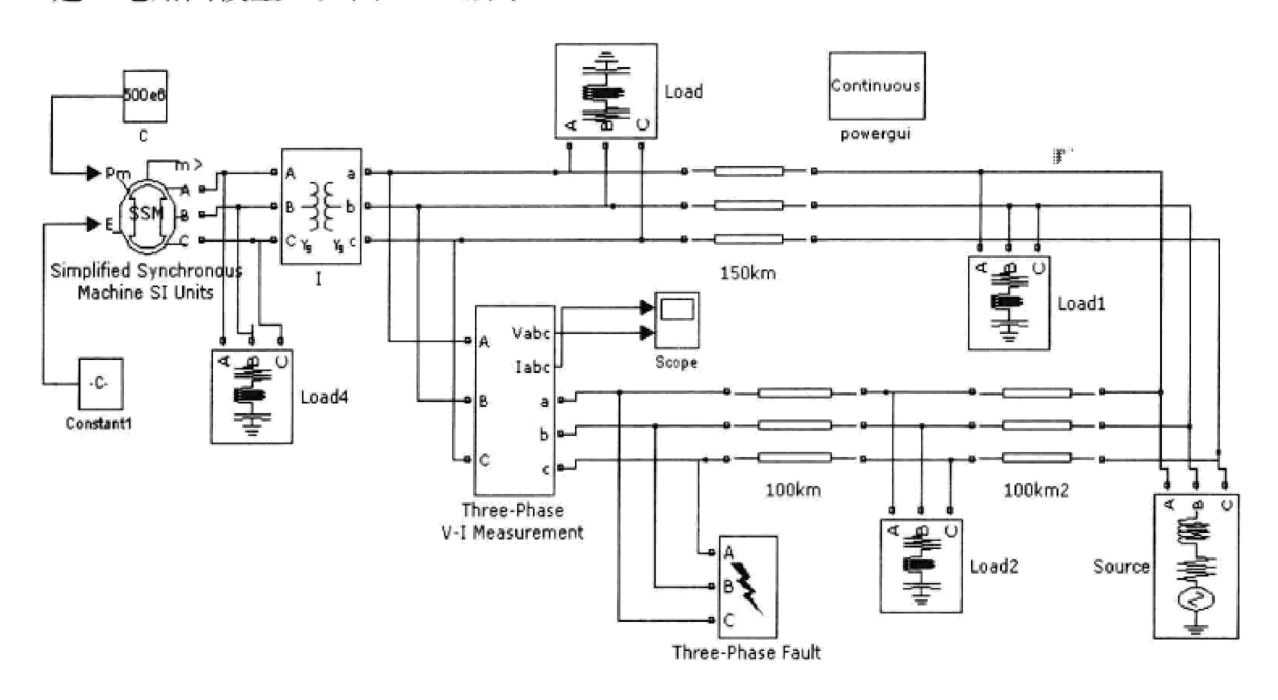


图 6-77 电路模型图

（2）模型参数设置

- 设置同步发电机参数，如图 6-78 所示。
- 设置三相变压器的参数，如图 6-79 所示。

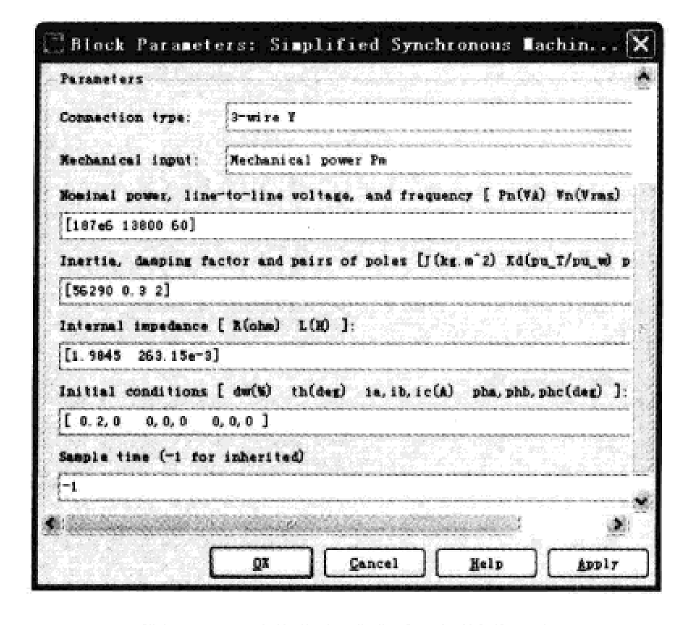


图 6-78 同步发电机的参数设置

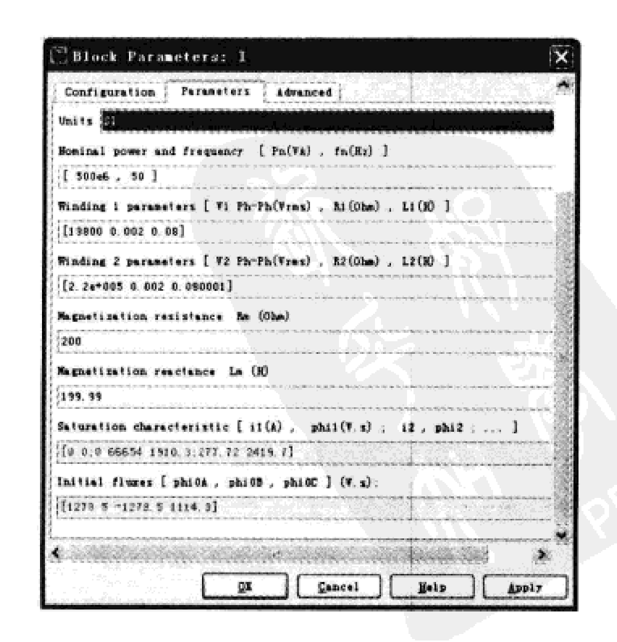


图 6-79 三相变压器的参数设置

● 设置 150km 分布参数电路参数，如图 6-80 所示。
● 设置 100km 分布参数电路参数，如图 6-81 所示。

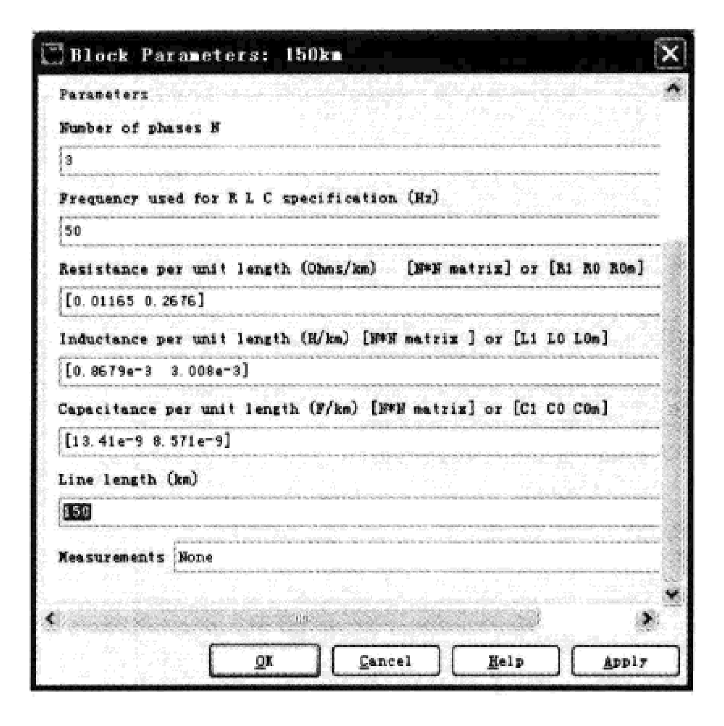


图 6-80 150km 分布参数电路的参数设置

图 6-81 100km 分布参数电路的参数设置

● 设置三相电压源参数，如图 6-82 所示。
● 设置三相串联 RLC 负载参数，如图 6-83 所示（其他 Load1、Load2 基本与其设置相同）。

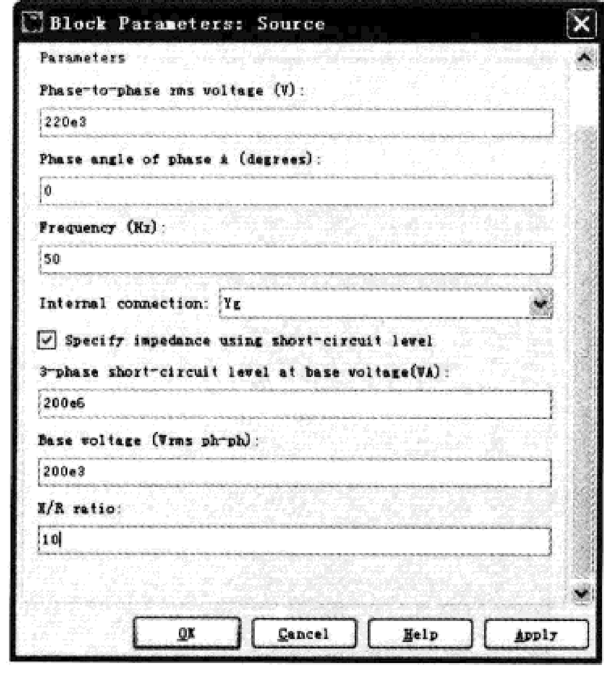


图 6-82 三相电压源的参数设置

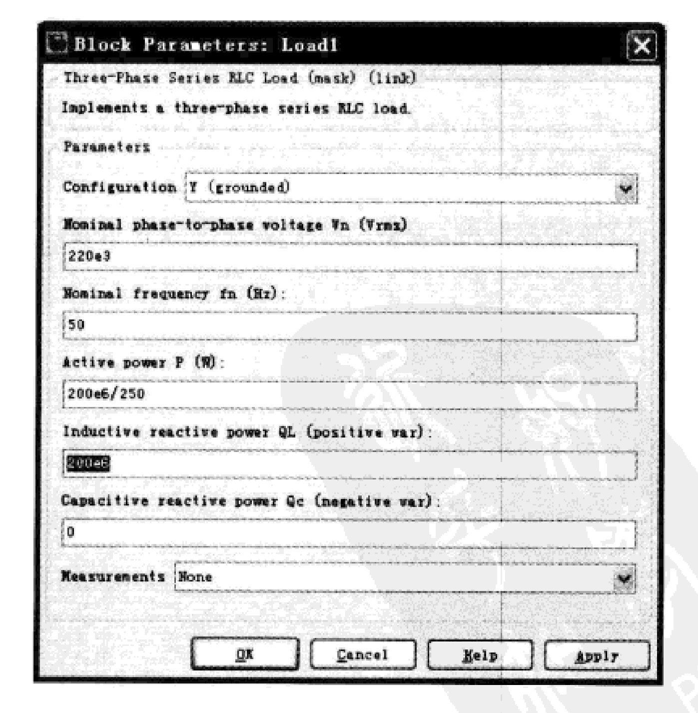


图 6-83 三相串联 RLC 负载 Load 的参数设置

● 设置三相串联 RLC 负载参数，Load4 如图 6-84 所示。
● 设置三相电压电流测量元器件参数，如图 6-85 所示。

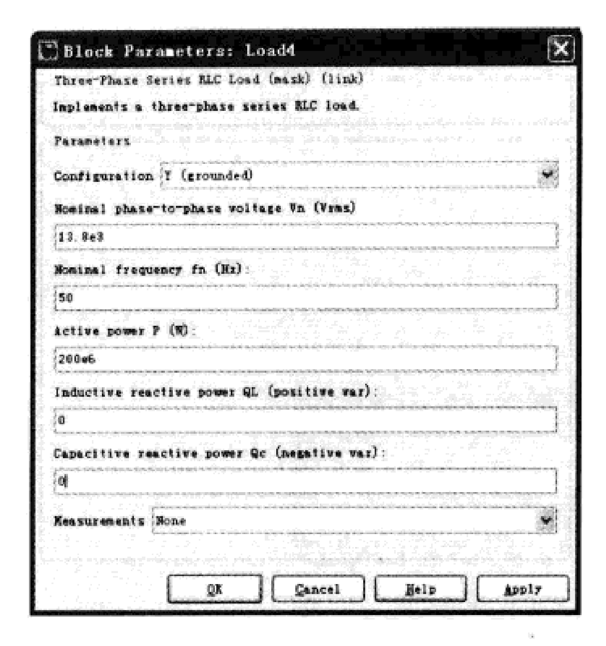


图 6-84 三相串联 RLC 负载 Load4 的参数设置

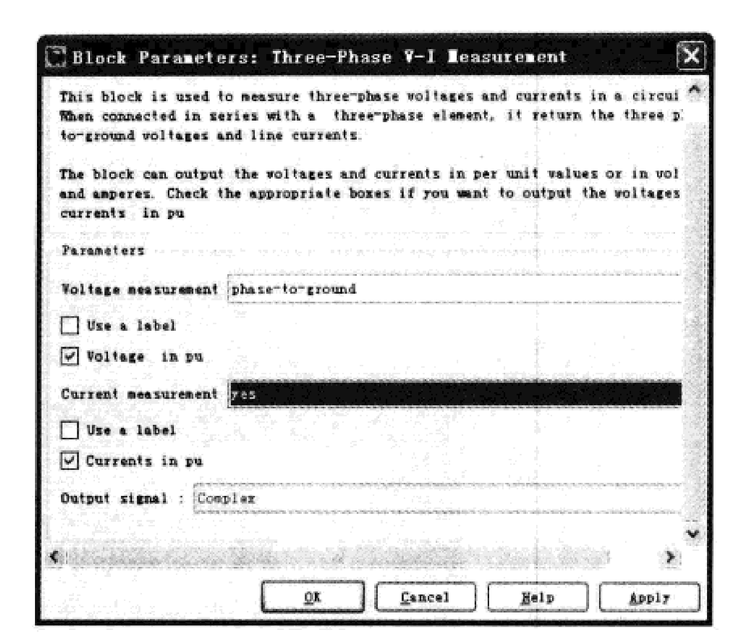


图 6-85 三相电压电流测量元器件的参数设置



● 设置三相短路元器件参数，如图 6-86 所示。

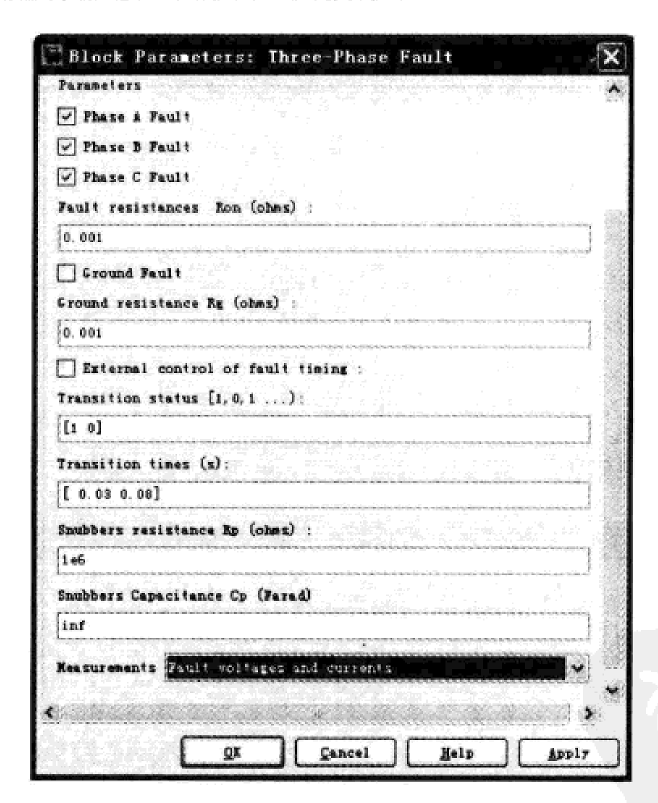


图 6-86 三相短路元器件参数设置

（3）仿真参数设置

对本系统仿真参数设置如下：

● Start time（开始时间）：0。
● Stop time（停止时间）：0.5。
● Type（求解程序类型）：Variable-step（可变步长），ode15s（stiff/NDF）。
● Max step size（最大步长）：auto（自动）。

- Min step size（最小步长）：auto（自动）。
- Initial step size（初始步长）：auto（自动）。
- Relative tolerance（相对容差）：1e-4。
- Absolute tolerance（绝对容差）：auto。

（4）仿真结果及分析

1）电路三相短路仿真分析。设置三相短路元器件参数为三相短路，得到电路三相短路时，母线 B1 的短路电压及电流波形，如图 6-87 所示。

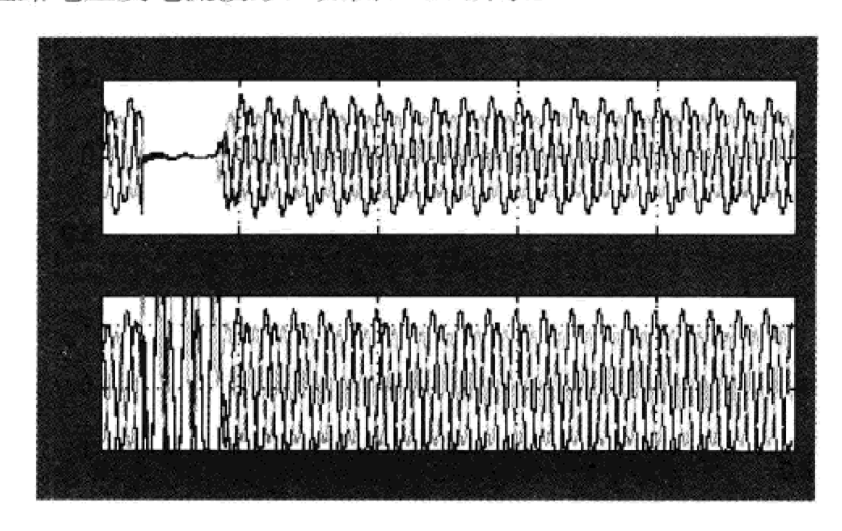


图 6-87　三相短路时的电压和电流

在 0～0.03s 时，电路工作在稳定状态，三相电流、电压对称。在 0.03s 时，发生三相短路，三相电压为 0，三相电流迅速上升为短路电流，并保持为三相对称，说明三相短路为对称短路。在 0.08s 时，故障切除，三相电压电流经暂态后达到新的稳定状态，并且重新恢复三相对称运行的工作状态。

2）线路两相相间短路分析。改变三相短路元器件参数为 A、B 两相相间短路，得到电路两相短路时，母线 B1 的短路电压及电流波形，如图 6-88 所示。

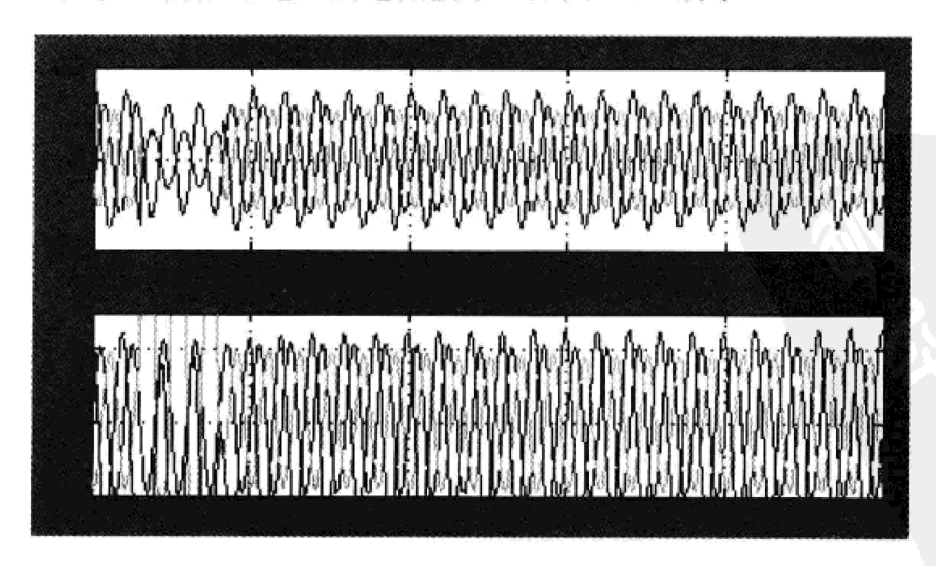


图 6-88　两相短路时的电压和电流

在 0～0.03s 时，电路工作在稳定状态，三相电流、电压对称。在 0.03s 时，发生 A、B

两相相间短路，A、B 两相电压减小，C 相电压基本保持不变；故障相 A、B 两相电流迅速上升为短路电流，C 相电流基本保持不变；三相电压、电流不再对称，说明两相短路为不对称短路。在 0.08s 时，故障切除，三相电压电流经暂态后达到新的稳定状态，并且重新恢复三相对称运行的工作状态。



3）线路两相接地短路分析。改变三相短路元器件参数 A、B 两相接地短路，得到电路两相接地短路时，母线 B1 的短路电压及电流波形，如图 6-89 所示。

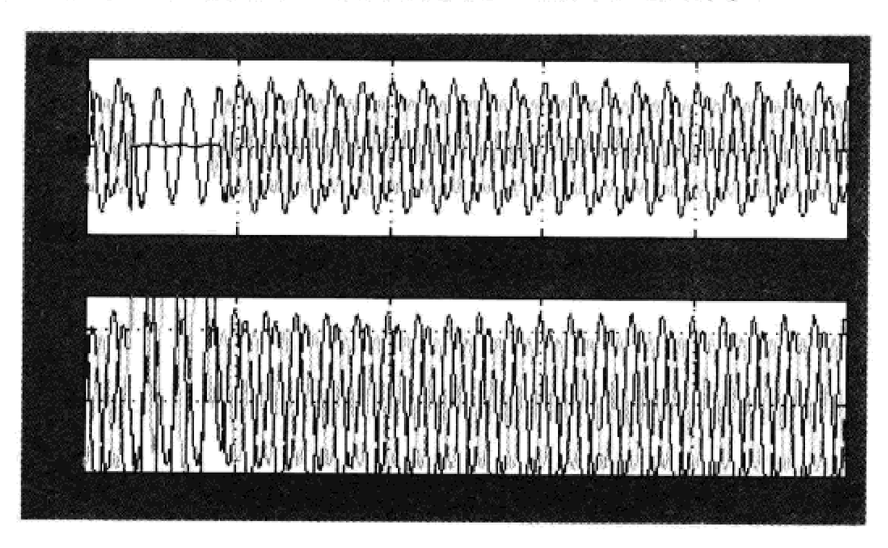


图 6-89　两相接地短路时的电压和电流

在 0～0.03s 时，电路工作在稳定状态，三相电流、电压对称。在 0.03s 时，发生 A、B 两相接地短路，A、B 两相电压基本为 0，C 相电压也相对减小；故障相 A、B 两相电流迅速上升为短路电流，C 相电流也相对增大；三相电压、电流不再对称，说明两相接地短路为不对称短路。在 0.08s 时，故障切除，三相电压电流经暂态后达到新的稳定状态，并且重新恢复三相对称运行的工作状态。

4）单相接地短路分析。改变三相短路元器件参数为 A 接地短路，得到电路单相接地短路时，母线 B1 的短路电压及电流波形，如图 6-90 所示。

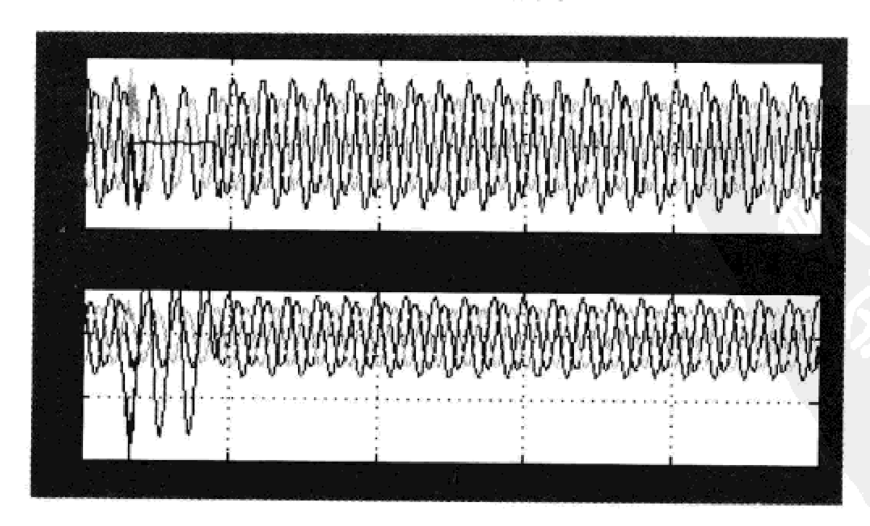


图 6-90　单相接地短路时的电压和电流

在 0～0.03s 时，电路工作在稳定状态，三相电流、电压对称。在 0.03s 时，发生 A 相接

地短路，A 相电压基本为 0，B、C 相电压也相对减小；故障相 A 相电流迅速上升为短路电流，B、C 相电流也相对增大；三相电压、电流不再对称，说明单相接地短路为不对称短路。在 0.08s 时，故障排除，三相电压电流经暂态后达到新的稳定状态，并且重新恢复三相对称运行的工作状态。

此外，在此电路图中加入相应的元器件，也可以对此电路图中的变量进行相序分析、相量图分析。





第 7 章　神经网络的仿真与分析

研究人类智能一直是科学发展中最有意义、最激动人心，也是最富有挑战性的课题。人工神经网络（Artificial Neural Network，ANN），简称为“神经网络”（NN），作为对人脑最简单的一种抽象和模拟，是探索人类智能奥秘的有力工具。神经网络技术作为智能科学的“领头羊”，是近年来发展起来的一门十分活跃的的交叉学科。它涉及生物、电子、计算机、数学、物理等学科，有着广泛的应用前景。

神经网络作为一种新的方法体系，具有分布并行处理、非线性映射、自适应学习和鲁棒容错等特性，这使得它在模式识别、控制优化、智能信息处理，以及故障诊断等方面都有广泛的应用。

目前，神经网络的研究可以分理论研究和应用研究两大方面：

1）利用神经生理与认知学研究大脑思维及智能机理。

2）利用神经科学基础理论的研究成果，用数理方法探索智能水平更高的人工神经网络模型，深入研究网络的算法和性能（如稳定性、收敛性、容错性和鲁棒性等），开发新的网络数理理论（如神经网络动力学、非线性神经场等）。

应用研究可分为以下两类：

1）神经网络的软件模拟和硬件实现的研究。

2）神经网络在各个领域中应用的研究，这些领域主要包括模式识别、信号处理、知识工程、专家系统、优化组合、智能控制等。

随着神经网络理论本身以及相关理论、相关技术的不断发展，神经网络的应用必将更加深入和广泛。

7.1　神经网络仿真概述

1．单输入单输出人工神经元

一个单输入单输出的人工神经元如图 7-1 所示。

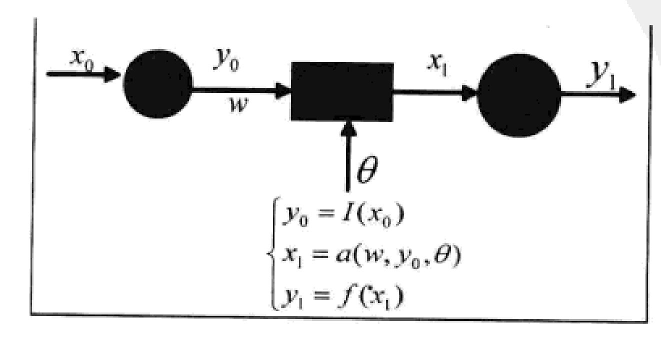


图 7-1　单输入单输出的人工神经元

从图 7-1 中可以看出，标量输入 x_0 经过预处理函数单元 I，预处理单元 I 的输出为

$$y_0 = I(x_0) \tag{7-1}$$

标有 a 的方框称为输入函数或激活函数，其输入/输出关系为：

$$x_1 = a(w, y_0, \theta) \tag{7-2}$$

式中，输入函数（激活函数）a 可以是任意的函数；w 称为权（或权值，在这里是标量）；θ 称为阈值。权值和阈值都是可以调整的参数。如果不想在神经元中使用阈值，可以忽略它。在后面的章节中将会出现这样的情况。输入函数的输出 x_1 通常被称为净输入，它被送入一个变换函数（或传输函数）f，在 f 中产生神经元的标量输出 y_1。

标有 f 的圆圈称为人工神经元的变换函数，其输入/输出关系为：

$$y_1 = f(x_1) \tag{7-3}$$

在具体应用时，人工神经元的预处理函数、输入函数和变换函数都有具体的形式。这里假定其预处理函数是恒等变换函数，输入函数是内积函数，如图 7-2 所示。

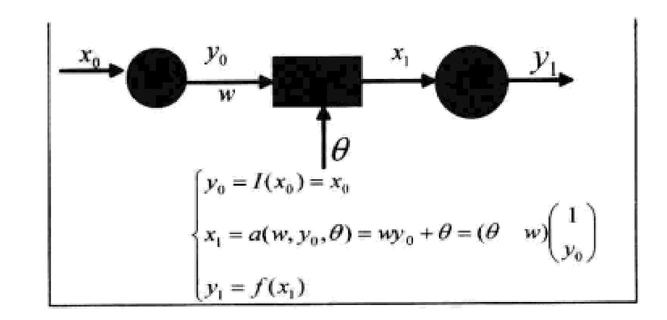


图 7-2　输入函数为内积函数时的单输入单输出人工神经元

根据图 7-2，预处理单元 I 的输入/输出关系为：

$$y_0 = I(x_0) = x_0 \tag{7-4}$$

输入函数（或激活函数）的输入/输出关系为：

$$x_1 = a(w, y_0, \theta) = wy_0 + \theta = \begin{pmatrix}\theta & w\end{pmatrix}\begin{pmatrix}1\\ y_0\end{pmatrix} \tag{7-5}$$

在本书中，变换函数 $y_1 = f(x_1)$ 的具体形式是类支集函数。

若将这个简单的模型和前面前面所讨论的生物神经元相对照，则权值 w 对应于突触的连接强度，细胞体对应于输入函数（激活函数）和变换函数，神经元输出 y_1 代表轴突信号。

在此使用样条函数神经元，引入了两类样条函数神经元。样条函数神经元结构简单，第一类样条函数神经元如图 7-3 所示。

x_0　$s(x_0)$　z_0　z_s

$$\begin{cases} z_0 = s(x_0) \\ z_s = \sum z_0 \end{cases}$$

图 7-3　第一类样条函数神经元结构

在图 7-3 所示的样条函数神经元结构图中，净输入：

$$z_0 = s(x_0) \tag{7-6}$$

式中，$s(x_0)$是自变量为x_0的一元样条函数，称为样条权函数。样条权函数的一个重要的特点就是它是输入自变量的函数，而不是传统方法中的常数。图 7-3 中标有 Add 的圆圈表示加法器，它交净输入相加求和，由于图 7-3 的净输入只有一个变量，则



$$z_s = \sum z_0 = z_0 \tag{7-7}$$

在此还引入另外一类样条函数神经元（第二类样条函数神经元），其结构如图 7-4 所示。

与图 7-3 不同，图 7-4 中标有 MUI 的圆圈表示乘法器，它将净输入相乘，由于图 7-4 的净输入只有一个变量，所以有：

$$z_s = \prod z_0 = z_0 \tag{7-8}$$

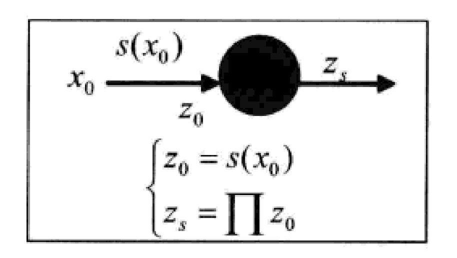


图 7-4 第二类样条函数神经结构

2. 多输入单输出人工神经元

通常，一个神经元有不止一个输入，具有 m 个输入的一般人工神经元如图 7-5 所示。

从图 7-5 中可以看出，每一个标量输入$x_{0i}(i=1,2,\cdots,m)$经过预处理函数单元 I 后，得到的输出为：

$$y_{0i} = I(x_{0i}) \tag{7-9}$$

标有 a 的方框是输入函数（或激活函数），其输入/输出关系为：

$$x_1 = a(w, y_0, \theta) \tag{7-10}$$

其中

$$w = (w_1, w_2, \cdots, w_m)^T \tag{7-11}$$

$$y = (y_{01}, y_{02}, \cdots, y_{0m})^T \tag{7-12}$$

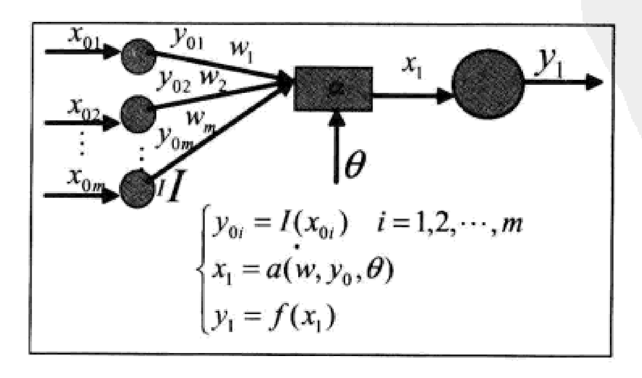


图 7-5 具有 m 个输入，一个输出的一般人工神经元

式（7-7）中输入函数（激活函数）a 可以是任意的函数，w 为权向量，θ 为阈值。净输入 x_1

被送入一个变换函数（或传输函数）f，在 f 中产生神经元的标量输出 y_1。除了输入端增加了维数之外，其余和图 7-1 是一样的。

与前面类似，这里假定其预处理函数是恒等变换函数，输入函数是内积函数，如图 7-6 所示。

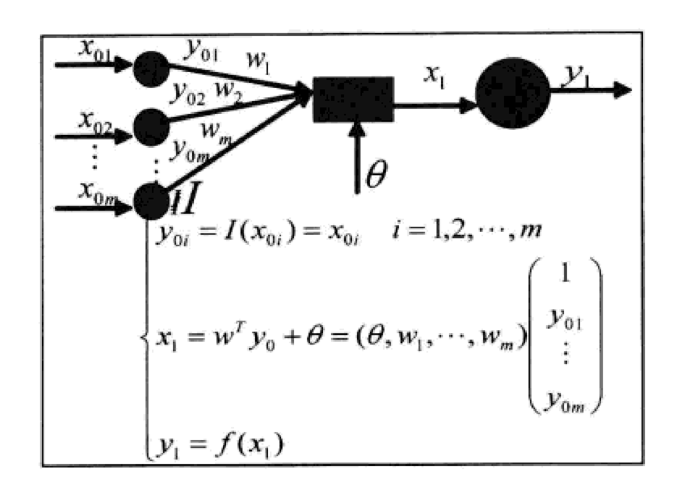


图 7-6　输入函数为内积函数时的多输入单输出人工神经元

根据图 7-6，预处理单元 I 的输入/输出关系为：

$$y_{0i}=I(x_{0i})=x_{0i} \quad i=1,2,\cdots,m \tag{7-13}$$

输入函数（或激活函数）的输入/输出关系为：

$$x_1=w^T y_0+\theta=(\theta,w_1,\cdots,w_m)\begin{pmatrix}1\\ y_{01}\\ \vdots\\ y_{0m}\end{pmatrix} \tag{7-14}$$

其中

$$\tilde{w}=(\theta,w_1,\cdots,w_m)^T \tag{7-15}$$

$$\tilde{y}_0=(1,y_{01},\cdots,y_{0m})^T \tag{7-16}$$

$\tilde{w}=(\theta,w_1,\cdots,w_m)^T$ 和 $\tilde{y}_0=(1,y_{01},\cdots,y_{0m})^T$ 分别称为扩充权和扩充输入。

与前面相同，变换函数 $y_1=f(x_1)$ 的具体形式是类支集函数。

第一类多输入单输出样条函数神经元结构如图 7-7 所示。

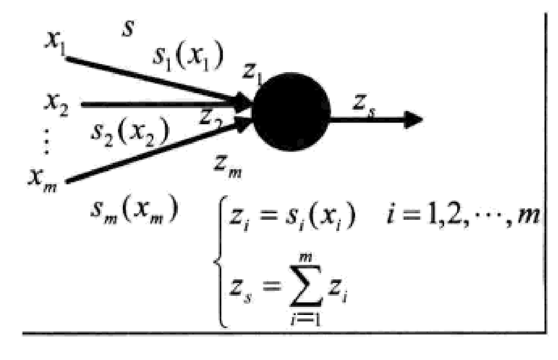


图 7-7　第一类多输入单输出样条函数神经元结构



在图 7-7 所示的第一类多输入单输出样条函数神经元结构中，净输入：

$$z_i = s_i(x_i), i = 1, 2, \cdots, m \tag{7-17}$$

式中，$s_i(x_i)$ 是自变量为 x_i 的一元样条函数，称为样条权函数。由于图 1-7 的净输入有 m 个变量，所以有：

$$z_s = \sum_{i=1}^{m} z_i \tag{7-18}$$

对于图 7-8 所示的第二类多输入单输出样条函数神经元结构，净输入仍然由式（7-17）计算，输出为：

$$z_s = \prod_{i=1}^{m} z_i \tag{7-19}$$

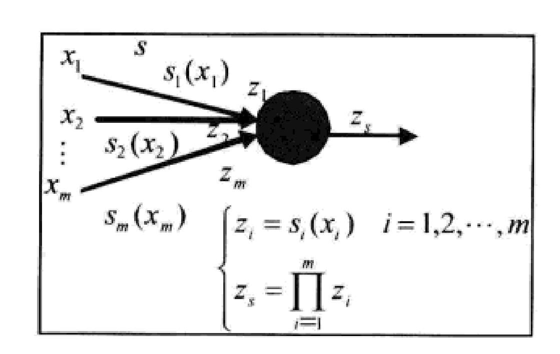


图 7-8　第二类多输入单输出样条函数神经元结构

7.2　线性神经网络仿真分析

7.2.1　线性神经网络应用函数

MATLAB R2009 的神经网络工具箱为线性网络提供了大量的函数，它们可分别用于线性网络的设计、创建、分析、训练及仿真等。下面对这些函数的功能、调用格式和注意的问题等进行展开介绍。

1. 线性网络创建和设计函数

（1）newlin 函数

功能：该函数可以创建一个线性层。所谓线性层是一个单独的层次，它的权函数为 dotprod，输入函数为 netsum，传递函数为 purelin。

其调用格式如下：

```
net = newlin(P,S,ID,LR)或 net = newlin(P,T,ID,LR)
```

其参数说明如下：

- net=newline：表示在一个对话框中创建一个新的网络。
- P：由 R 个输入元素的最大值和最小值组成的 $R\times2$ 维矩阵。
- S：输出向量的数目。

- ID：输入延迟向量，默认为[0]。
- LR：学习速率，默认为 0.01。
- net：函数返回值，一个新的线性层。

【例 7-1】 newlin 函数示例。

```
>> net = newlin([-1 1],1,[0 1],0.01);
P1 = {0 -1 1 1 0 -1 1 0 0 1};
Y = sim(net,P1);
T1 = {0 -1 0 2 1 -1 0 1 0 1};
[net,Y,E,Pf] = adapt(net,P1,T1); Y
```

运行程序，输出结果如下：

```
Y =
 [0]  [0]   [0]   [0]   [0.0300]   [-0.0103]  [0.0200]  [0.0395]  [0.0192]  [0.0587]
```

（2）newlind 函数

功能：该函数可以设计一个线性层，它通过输入向量和目标向量来计算线性层的权值和阈值。

其调用格式如下：

```
net = newlind(P,T,Pi)
```

其参数说明如下：

- net=newlind：表示在一个对话框中创建一个新的网络。
- P：Q 组输入向量组成的 $R\times Q$ 维矩阵。
- T：Q 组目标分类向量组成的 $S\times Q$ 维矩阵。
- Pi：初始输入延迟状态的 ID 个单元阵列，每个元素 Pi{i, k}都是一个有最小值。
- net：函数返回值，一个线性层，它的输出误差平方和对于输入 P 来说具有最小值。

【例 7-2】 newlind 函数示例。

```
>> P1 = {1 2 1 3 3 2}; Pi1 = {1 3 0};
P2 = {1 2 1 1 2 1}; Pi2 = {2 1 2};
T1 = {5.0 6.1 4.0 6.0 6.9 8.0};
T2 = {11.0 12.1 10.1 10.9 13.0 13.0};
net = newlind([P1; P2],[T1; T2],[Pi1; Pi2]);
Y = sim(net,[P1; P2],[Pi1; Pi2]);
Y1 = Y(1,:)
Y2 = Y(2,:)
```

运行程序，输出结果如下：

```
Y1 =
    [5.0000]    [6.1000]    [4]    [6]    [6.9000]    [8.0000]
Y2 =
    [11.0000]    [12.1000]    [10.1000]    [10.9000]    [13.0000]    [13]
```



2．学习函数

（1）learnwh 函数

功能：该函数为 Widrow-Hoff 学习函数，也称为 delta 准则或最小方差准则学习函数。它可以修改神经元的权值或阈值，使输出误差的平方和最小。

其调用格式如下：

```
[dW,LS] = learnwh(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
[db,LS] = learnwh(b,ones(1,Q),Z,N,A,T,E,gW,gA,D,LP,LS)
info = learnwh(code)
```

其参数说明如下：

- W：$S\times R$ 维的加权矩阵（或为 $S\times 1$ 的阈值矩阵）。
- P：Q 组 R 维的输入向量（或为 Q 个单值输入）。
- Z：Q 组 S 维的加权输入向量。
- N：Q 组 S 维的网络输入向量。
- A：Q 组 S 维的输出向量。
- T：Q 组 S 维的目标向量。
- E：Q 组 S 维的误差向量。
- gW：$S\times R$ 维的性能参数的梯度。
- gA：Q 组 S 维的性能参数的输出梯度。
- D：神经元距离。
- LP：学习参数，若没有则为空。
- LS：学习状态，初始值为空。
- dW：$S\times R$ 维权值（或阈值）的变化矩阵。
- LS：新的学习状态。
- learnwh(code)：针对不同的 code 返回相应的有用信息，包括 pnames——返回学习参数的名称；pdefaults——返回默认的训练参数。Needg——如果函数使用了 gW 或 gA，则返回 1。

【例 7-3】 learnwh 函数示例。

```
>> p = rand(2,1);
e = rand(3,1);
lp.lr = 0.5;
dW = learnwh([],p,[],[],[],[],e,[],[],[],lp,[])
```

运行程序，输出结果如下：

```
dW =
    0.0517    0.0575
    0.3721    0.4137
    0.2576    0.2864
```

（2）maxlinlr 函数

功能：该函数为分析函数，用于计算线性层的最大学习速率。

其调用格式如下：

```
lr = maxlinlr(P)
lr = maxlinlr(P,'bias')
```

其参数说明如下：

- P：输入向量的 $R \times Q$ 维矩阵。
- lr = maxlinlr(P)：针对不带阈值的线性层得到一个所需要的最大学习速率。
- lr = maxlinlr(P,'bias')：针对带有阈值的线性层得到一个所需要的最大学习速率。

【例 7-4】 maxlinlr 函数示例。

```
>> P = [1 2 -4 7; 0.1 3 10 6];
lr = maxlinlr(P,'bias')
```

运行程序，输出结果如下：

```
lr =
    0.0067
```

7.2.2 线性神经网络仿真设计分析

【例 7-5】 以自适应线性网络实现噪声对消。

1）问题分析。对于一个最优的滤波器，希望通过滤波将信号中的噪声去掉，这对一般的滤波器很难完全做到。利用自适应线性网络实现噪声对消的原理框图如图 7-9 所示。

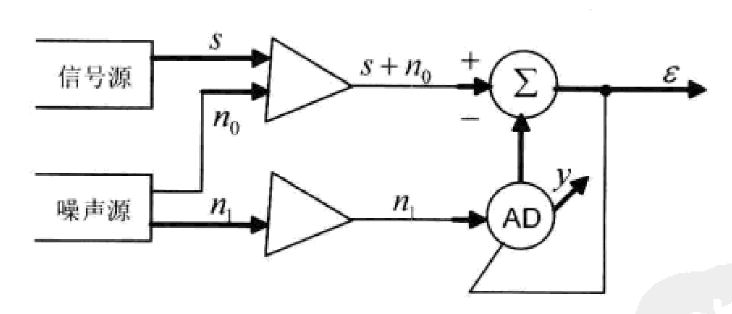


图 7-9 噪声对消原理框图

图中 s 为原始输入信号，假设为平稳的零均值随机信号；n_0 为与 s 不相关的随机噪声；n_1 为与 n_0 相关的信号；系统输出为 ε；$s+n_0$ 为 ADALINE 神经元的预期输出，y 为 ADALINE 神经元的输出。则

$$\varepsilon = s + n_0 - y$$

$$E[\varepsilon^2] = E[(s+n_0-y)^2] = E[s^2] + 2E[s \cdot (n_0 - y)] + E[(n_0-y)^2] = E[s^2] + E[(n_0-y)^2]$$

通过 ADALINE 调节，得到：

$$E_{\min}[\varepsilon^2] = E_{\min}[s^2] + E_{\min}[(n_0-y)^2]$$

式中，当 $E_{\min}[(n_0 - y)^2] \to 0$ 时，$y \to n_0$，其输出 ε 为 s，则噪声被抵消。

采用这种系统来完成对胎儿心率的检测，可以得到十分满意的结果。由于测量胎儿的心率一定会受到母体心率的干扰，而且母亲心率很强，但与胎儿心率是相互独立的，所以可将母体心率作为噪声源 n_1 输入 ADALINE 中，混有噪声的胎儿心率信号作为目标响应，通常对消后，系统就可以得到清晰的胎儿心率。

这里，假设传输信号为正弦波信号，噪声为随机噪声，进行自适应线性神经网络设计。

2）实现噪声对消的自适应线性神经元的输入向量为随机噪声 n_l；正弦波信号与随机噪声之和为 ADALINE 神经元的目标向量；输出信号为网络调整过程中的误差信号。其实现的 MATLAB 程序代码如下：

```
>> clear all;
%定义输入向量和目标向量
time=0.01:0.01:10;                %时间变量
noise=(rand(1,1000)-0.5)*4;       %随机噪声
input=sin(time);                  %信号
p=noise;                          %将噪声作为 ADALINE 的输入向量
t=input+noise;                    %将噪声+信号作为目标向量
%创建线性神经网络
net=newlin([-1 1],1,0,0.0005)
%线性神经网络的自适应调整(训练)
net.adaptParam.passes=70;
%输出信号 output 为网络调整过程中的噪声
[net,y,output]=adapt(net,p,t);
%绘制信号,迭加随机噪声的信号,输出信号的波形
hold on;
%绘制信号的波形
subplot(3,1,1);
plot(time,input,'r');
xlabel('t','position',[10.5,-1]);
ylabel('信号波形 sin(t)','fontsize',9);
subplot(3,1,2);
%绘制迭加随机噪声的信号波形
plot(time,t,'m');
xlabel('t','position',[10.5,-5]);
ylabel('随机信号波形 sin(t)+noise(t)','fontsize',9);
%绘制输出信号的波形
subplot(3,1,3);
plot(time,output,'y');
xlabel('t','position',[10.5,-2]);
ylabel('输出信号波形 y(t)','fontsize',9);
```



运行程序，输出网络信息如下：

```
net =
```

```
Neural Network object:
architecture:
        numInputs: 1
        numLayers: 1
      biasConnect: [1]
     inputConnect: [1]
     layerConnect: [0]
    outputConnect: [1]
       numOutputs: 1  (read-only)
 numInputDelays: 0  (read-only)
 numLayerDelays: 0  (read-only)
subobject structures:
           inputs: {1x1 cell} of inputs
           layers: {1x1 cell} of layers
          outputs: {1x1 cell} containing 1 output
           biases: {1x1 cell} containing 1 bias
     inputWeights: {1x1 cell} containing 1 input weight
     layerWeights: {1x1 cell} containing no layer weights
functions:
             adaptFcn: 'trains'
            divideFcn: (none)
          gradientFcn: 'calcgrad'
              initFcn: 'initlay'
           performFcn: 'mse'
             plotFcns: {'plotperform','plottrainstate'}
             trainFcn: 'trainb'
parameters:
          adaptParam: .passes
         divideParam: (none)
       gradientParam: (none)
           initParam: (none)
        performParam: (none)
          trainParam: .show, .showWindow, .showCommandLine, .epochs,
                      .time, .goal, .max_fail
weight and bias values:
                  IW: {1x1 cell} containing 1 input weight matrix
                  LW: {1x1 cell} containing no layer weight matrices
                   b: {1x1 cell} containing 1 bias vector
other:
                name: ''
            userdata: (user information)
```

运行程序，输出效果图如图 7-10 所示。

【例 7-6】 实现自适应预测的线性网络。设自适应滤波器如图 7-11 所示，该滤波器的目的是要从输入信号的前两个时刻的值预测当前时刻的值。

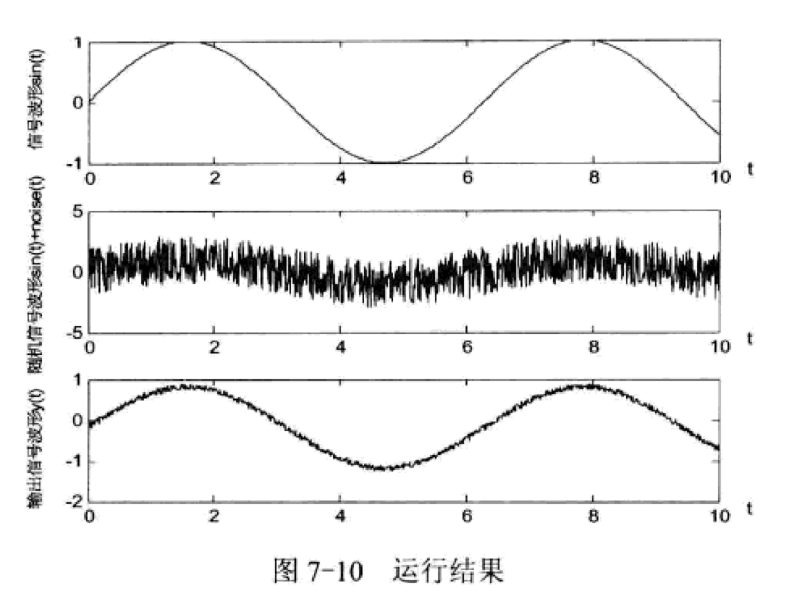


图 7-10 运行结果

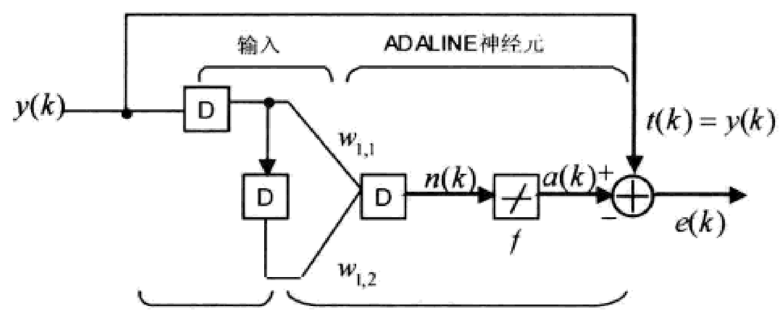


图 7-11 自适应滤波器模型

图中 D 为延迟单元，多个延迟单元可以构成抽头延迟线，如图 7-12 所示。

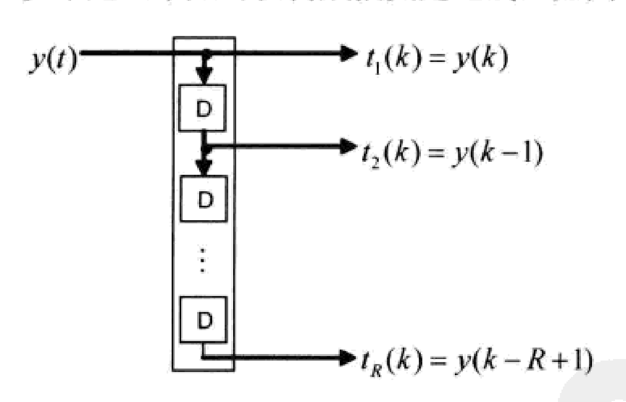


图 7-12 抽头延迟线

设输入信号为一随机序列，试编写 MATLAB 程序，画出上述自适应滤波器的输入/输出波形。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%定义输入向量和目标向量
time=0.5:0.5:20;            %时间变量
y=(rand(1,40)-0.5)*4;       %定义随机输入信号
p=con2seq(y);               %将随机输入向量转换为串行向量
delays=[1 2];               %定义 ADALINE 神经元输入延迟量
t=p;                        %定义 ADALINE 神经元的目标向量
```



```
%创建线性神经网络
net=newlin(minmax(y),1,delays,0.0005);
%线性神经网络的自适应调速(训练)
net.adaptParam.passes=70;
%输出信号 output 为网络调整过程中的误差
[net,a,output]=adapt(net,p,t);
%绘制随机信号,输出信号的波形
hold on;
%绘制信号的波形
subplot(3,1,1);
plot(time,y,'k*-')
xlabel('t','position',[20.5,-1.8]);
title('随机输入信号 sin(t)');;
axis([0 20 -2 2]);
subplot(3,1,2);
output=seq2con(output);
%绘制预测输出信号的波形
plot(time,output{1},'ko-');
xlabel('t','position',[20.5,-1.8]);
title('预测输出信号 y(t)');
axis([0 20 -2 2]);
%绘制输出信号的波形
subplot(3,1,3);
e=output{1}-y;
plot(time,e,'r--');   %绘制误差曲线
xlabel('t','position',[20.5,-1.8]);
title('误差曲线 e(t)');
axis([0 20 -2 2]);
hold off;
```

其运行结果如图 7-13 所示。从图中可以看出，输出信号波形与输入信号波形基本一致，误差较小，输出波形较好地预测了输入波形。

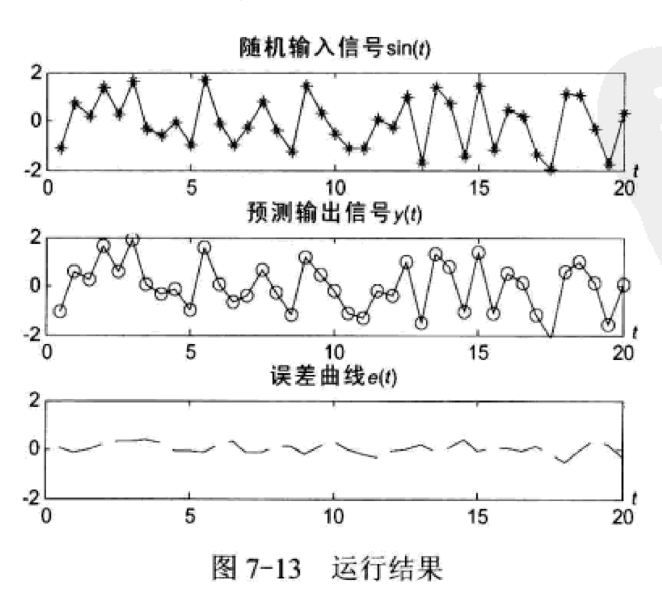


图 7-13　运行结果

值得一提的是，在程序设计中，需要注意学习率和训练步长的选择。学习率过大，学习的过程将不稳定，且误差会更大；学习率过小，学习的过程将变慢，需要的训练步长数将加大。选择不当，将得不到满意的结果。



7.3 感知器网络仿真分析

7.3.1 感知器神经网络应用函数

MATLAB R2009 的在线帮助功能丰富，在 MATLAB 工作空间的命令行输入 help percept，即可得到所有与感器（Perceptron）相关的函数，进一步利用 help 功能又能得到相关函数的详细介绍。

1. 创建函数（newp）

功能：可通过感知器生成函数来创建一个感知器，并且可对感知器进行初始化、仿真和训练等。

其调用格式如下：

```
net = newp(PR,S,TF,LF)
```

其参数说明如下：

- net：函数返回参数，表示生成的感知器的网络。
- PR：代表 $R\times Q$ 维的输入向量。
- S：代表 $S\times Q$ 维的目标向量。
- TF：感知器的传递函数，可选参数为 hardlim 和 hardlims，默认为 hardlim。
- LF：感知器的学习函数，可选参数为 learnp 和 learnpn，默认为 learnp。

【例 7-7】 newp 示例。

```
>> net = newp([0 1; -2 2],1);
P1 = {[0; 0] [0; 1] [1; 0] [1; 1]};
Y = sim(net,P1)
T1 = {0 0 0 1};
net.adaptParam.passes = 10;
net = adapt(net,P1,T1);
Y = sim(net,P1)
P2 = [0 0 1 1; 0 1 0 1];
T2 = [0 1 1 1];
net = init(net);
Y = sim(net,P2)
net.trainParam.epochs = 20;
net = train(net,P2,T2);
Y = sim(net,P2)
```

运行程序，输出结果如下：

```
Y =
```

```
     [1]    [1]    [1]    [1]
Y =
     [0]    [0]    [0]    [1]
Y =
      1      1      1      1
Y =
      0      1      1      1
```

2. 显示函数

（1）plotpc 函数

功能：该函数用于在感器向量图中绘制分界线。

其调用格式如下：

```
plotpc(W,B)
plotpc(W,B,H)
```

其参数说明如下：

- W：$S \times R$ 维的加权矩阵（R 必须小于等于 3）。
- B：$S \times 1$ 维的阈值向量。
- H：最后画线的控制权。
- plotpc(W,B)：返回的是对所绘制分界线的控制权。
- plotpc(W,B,H)：用于在绘制新线之前检查最新绘制的分界线。

（2）plotpv 函数

功能：该函数用于绘制感知器的输入向量和目标向量。

其调用格式如下：

```
plotpv(P,T)
plotpv(P,T,V)
```

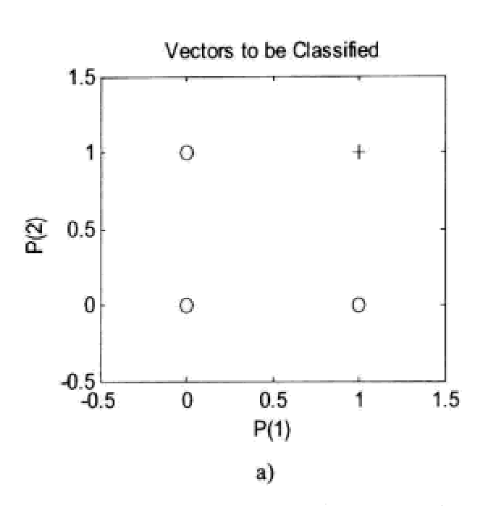
其参数说明如下：

- P：Q 组 R 维的输入向量。
- T：Q 组 R 维的双目标向量。
- V=[x_min x_max y_min y_max]：图形的最大值，绘制工作必须位于 V 所限定的范围中。
- plotpv(P,T)：以 T 为标尺，绘制 P 的列向量。
- plotpv(P,T,V)：在 V 的范围中绘制 P 的列向量。

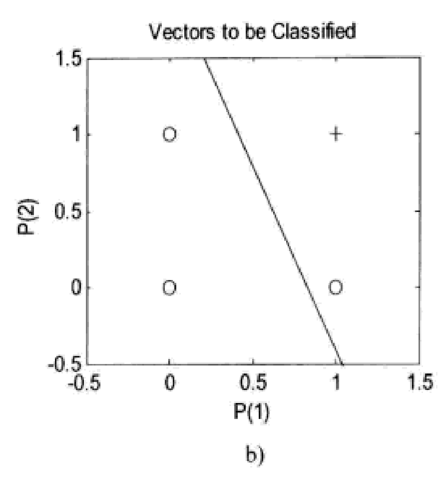
【例 7-8】 plotpc 与 plotpv 函数示例。

```
>> p = [0 0 1 1; 0 1 0 1];
t = [0 0 0 1];
plotpv(p,t)
net = newp(minmax(p),1);
net.iw{1,1} = [-1.2 -0.5];
net.b{1} = 1;
plotpc(net.iw{1,1},net.b{1})
```

运行程序，输出效果如图 7-14 所示。



a)



b)



图 7-14　显示函数绘制结果

a) plotpv(p,t)函数绘制结果　b) plotpc 函数绘制结果

（3）性能函数（mae）

功能：该函数以平均绝对误差为准则，确定神经网络性能的函数。

其调用格式如下：

```
perf = mae(E,Y,X,FP)
dPerf_dy = mae('dy',E,Y,X,perf,FP)
dPerf_dx = mae('dx',E,Y,X,perf,FP)
info = mae(code)
```

其参数说明如下：

- E：误差向量矩阵（或向量）。
- X：所有的权值和阈值向量，可忽略。
- FP：待评定的神经网络。
- perf：函数返回值，为平均绝对误差。
- info = mae(code)：根据 code 值的不同，返回不同的信息，包括 derive——返回导数函数的名称；name——返回函数全称；pnames——返回训练参数的名称。

【例 7-9】 性能函数示例。

```
>> net = newp([-10 10],1);
p = [-10 -5 0 5 10];
t = [0 0 1 1 1];
y = sim(net,p);
e = t-y
perf = mae(e)
```

运行程序，输出结果如下：

```
e =
     -1     -1     0     0     0
perf =
     0.4000
```

7.3.2 感知器神经网络仿真设计分析

1．单层感知器神经网络设计的基本方法

单层感器神经网络的 MATLAB 仿真程序设计主要包括以下几个方面。

（1）以 newp 创建感知器神经网络

首先根据所要解决的问题，确定输入向量的取值范围和维数、网络层的神经元数目、传递函数和学习函数等；然后以单层感知器神经网络的创建函数 newp 创建网络。

（2）以 train 训练创建网络

构造训练样本集，确定每个样本的输入向量和目标向量，调用 train 函数对网络进行训练，并根据训练的情况决定是否训练参数；以得到满足误差性能指标的神经网络，然后进行存储。

（3）以 sim 对训练后的网络进行仿真

构造训练样本集，加载训练后的网络，调用 sim 函数，以测试样本集进行仿真，检查网络的性能。

从以上过程可以看出，重要的感知器神经网络函数有 newp、train 和 sim，除此之外还涉及 init、trainc、dotpord、netsum、mae、plotpc、plotpv 等。

2．多层感知器神经网络的设计方法

单层感知器由于其结构和学习规则上的局限性，其应用也受到一定的限制，即它只能对线性可分的向量集合进行分类。为了解决线性不可分的输入向量的分类问题，可以增加网络层。

由于感知器神经网络学习规则的限制，它只能对单层感知器神经网络进行训练，那么如何进行多层感知器神经网络的设计呢？这里提供一种二层感知器神经网络的设计方法。

1）把神经网络的第一层设计为随机感知器层，且不对它进行训练，而是随机初始化它的权值和阈值，当它接收各输入元素的值时，其输出也是随机的。但其权值和阈值一旦固定下来，对输入向量模式的映射也随之确定下来。

2）以第一层的输出作为第二感知器层的输入，并对应输入模式，确定第二感知器层的目标函数向量，然后对第二感知器层进行训练。

3）由于第一随机感知器层的输出是随机的，所以在训练过程中，整个网络可能达到训练误差性能指标，也可能达不到训练误差性能指标。所以，当达不到训练误差指标，需要重新对随机感知器层的权值和阈值进行初始化赋值，可以将其初始化函数设置为随机函数，然后用 init 函数重新初始化。程序一次运行的结果往往达不到设计要求，需要反复运行，直至达到要求为止。

下面对感知器神经网络进行 MATLAB 仿真程序的设计。

【例 7-10】 本例我们将说明奇异输入样本矢量对感知器神经网络训练结果的影响。所谓奇异样本，是指相对于其他输入样本特别大或特别小的样本矢量。奇异样本矢量的存在会使基于常规感知器学习规则 learnp 的网络训练效率下降。其样本矢量点如下：

输入矢量为：$p=\begin{pmatrix} -0.5 & -0.5 & 0.3 & -0.1 & -40 \\ -0.5 & 0.5 & -0.5 & 1.0 & 100 \end{pmatrix}$

目标分类矢量为：$t=[1 \quad 1 \quad 0 \quad 0 \quad 1]$

由于样本点[-40; 100]明显不同于其他输入样本矢量，因此它是奇异样本点。奇异样本点的加入使得感知器网络的训练时间大大加长。其实现的 MATLAB 程序代码如下：



```
clear all;
% P 为输入矢量
P=[-0.5,-0.5,0.3,-0.1,-40;-0.5,0.5,-0.5,1,100];
%T 为目标矢量
T=[1 1 0 0 1];
%绘制待分类的数据点图
figure(1);plotpv(P,T);
%创建待分类的数据点图
net=newp(minmax(P),1);
line=plotpc(net.IW{1},net.b{1});
%训练感知器神经网络
e=1;n=0;
while(sse(e))
    [net,y,e]=adapt(net,P,T);
    n=n+1;
    perf(n)=sse(e);
    line=plotpc(net.IW{1},net.b{1},line);
end
%绘制误差变化曲线
figure(2);plot(perf);
%利用训练好的感知器网络进行分类
p=[-20;20];   %新的数据点
a=sim(net,p);
figure(3);plotpv(p,a);   %绘制新的数据点
%绘制新的分类结果
hold on;
plotpv(P,T);
figure(4)
plotpc(net.IW{1},net.b{1});
```

运行程序，输出效果如图 7-15 所示。

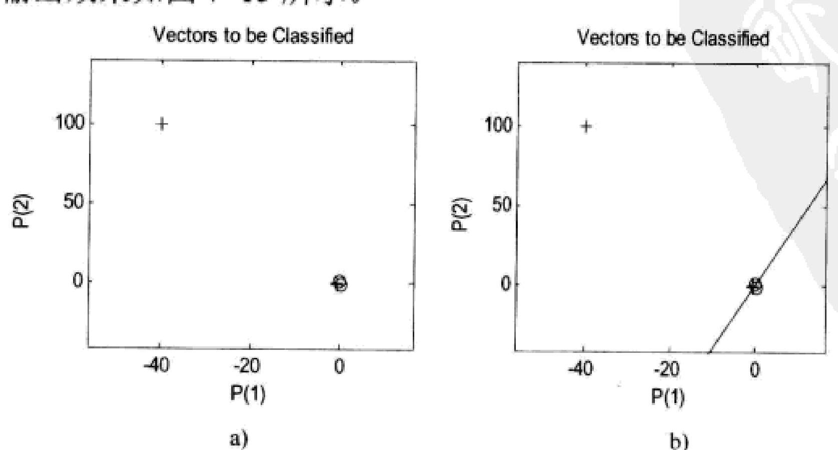


图 7-15　感知器神经网络对奇异矢量的分类结果

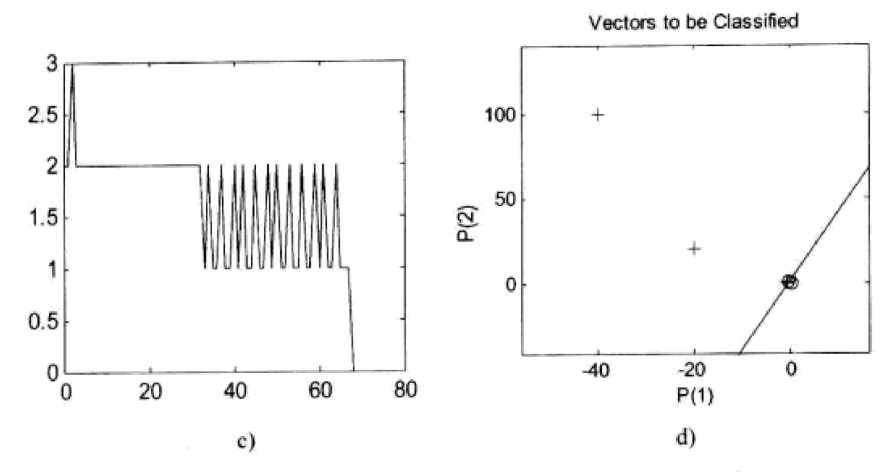


图 7-15 感知器神经网络对奇异矢量的分类结果（续）

a) 待分类的数据点 b) 分类结果 c) 误差变化曲线图 d) 新的分类结果

【例 7-11】 单层感知器网络不能模拟异或函数的问题，这里用两层感知器神经网络来实现。

（1）问题分析

异或问题真值表见表 7-1。

表 7-1 异或问题真值表

输入 p1	输入 p2	输出 a
0	0	0
1	0	1
0	1	1
1	1	0

若把异或问题看成 $p_1 - p_2$ 平面上的点，则点 $A_0(0,0)$，$A_1(1,1)$ 表示输出为 0 的两个点，$B_0(0,0)$，$B_1(1,1)$ 表示输出为 1 的两个点，如图 7-16 所示。

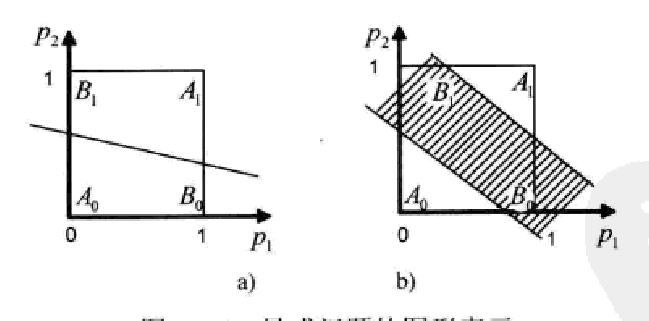


图 7-16 异或问题的图形表示

a) 单层感知器的超平面划分 b) 多层感知器的超平面划分

从图中可以看出，无论在平面上怎样用一条直线也不可能将输出为 0 和 1 的两种模式分开，而用两条直线就能将输出为 0 和 1 的两种模式分开。

（2）神经网络的设计

根据以上分析，如果用两层感知器，每层感知器可以构成一条直线划分，则可以解决模拟异或函数的问题。以图 7-17 所示两层神经网络来实现，其中隐层为随机感知器层（net1），神经元数目设计为 3，其权值和阈值是随机的，它的输出作为输出层（分类层）的

输入；输出层为感知器层（net2），其神经元数为 1，这里仅对该层进行训练。

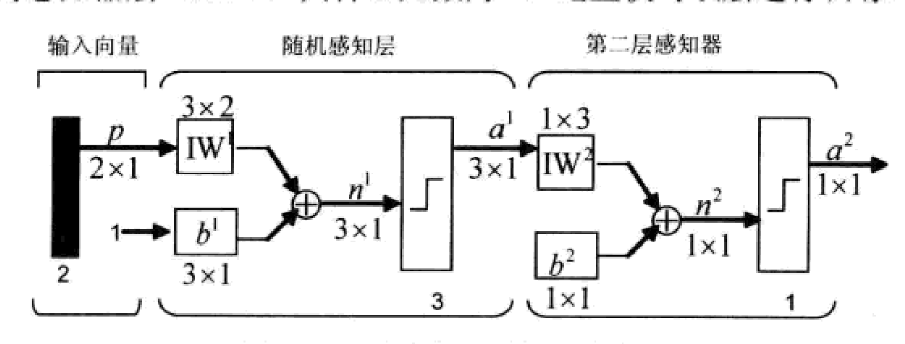
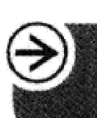


图 7-17 两层感知器神经网络模型



（3）多层感知器神经网络的 MATLAB 实现

其代码如下。

```
clear all;
%初始化随机感知器层
PR1=[0 1;0 1];
net1=newp(PR1,3);
net1.inputweights{1}.initFcn='rands';
net1.biases{1}.initFcn='rands';
net1=init(net1);
IW1=net1.iw{1}
B1=net1.b{1}
%随机感知器层仿真量
P1=[0 0;0 1;1 0;1 1]';
[A1,Pf]=sim(net1,P1);
%初始化第二感知器层
PR2=[0 1;0 1;0 1];
net2=newp(PR2,1);
%训练第二感知器层
net2.trainParam.epochs=10;
net2.trainParam.show=1;
P2=ones(3,4);
P2=P2.*A1;
T2=[0 1 1 0];
[net2,TR2]=train(net2,P2,T2);
epoch2=TR2.epoch
perf2=TR2.perf
IW2=net2.iw{1}
B2=net2.b{1}
%存储训练后的网络
save net34 net1 net2
```

因为随机感知器层的输出是随机的，所以整个网络可能达到训练误差指标，也可能达不到训练误差指标。因此，当达不到训练误差指标时，需要重新对随机感知器层的权值和阈值进行初始化赋值，在本例程序中，是通过将其初始化函数设置为随机函数，然后用 init 函数重新初始化来实现的。该程序一次运行的结果可能达不到设计要求，需要反复运行，直至达

到要求为止。正因为如此，每次训练得到的结果也不尽相同。

这样做是因为设计受到感知器的学习算法的限制，对于多层网络，当然有更好的学习算法，比如后面将要介绍的 BP 网络学习算法。

其中达到训练误差指标的一种运行结果如下：

```
IW1 =
    0.7200    -0.0069
    0.7073     0.7995
    0.1871     0.6433
B1 =
   -0.6983
    0.3958
   -0.2433
TRAINC, Epoch 0/10
TRAINC, Epoch 1/10
TRAINC, Epoch 2/10
TRAINC, Epoch 3/10
TRAINC, Epoch 4/10
TRAINC, Epoch 5/10
TRAINC, Epoch 6/10
TRAINC, Epoch 7/10
TRAINC, Epoch 8/10
TRAINC, Epoch 9/10
TRAINC, Epoch 10/10
TRAINC, Maximum epoch reached.
epoch2 =
     0     1     2     3     4     5     6     7     8     9    10
perf2 =
    0.5000    0.7500    0.7500    0.5000    0.5000    0.7500    1.0000    1.0000
    0.2500    0.5000    0.5000
IW2 =
    -3     2    -2
B2 =
     2
```

训练误差性能曲线如图 7-18 所示。

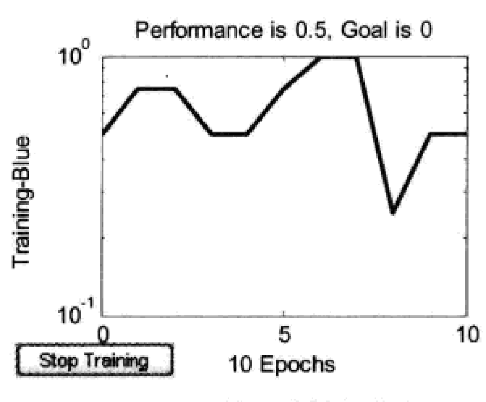


图 7-18　训练误差性能曲线

下面为多层感知器神经网络仿真的 MATLAB 程序。

```
clear all;
%加载训练后的网络
load net34 net2
%随机感知器层仿真
P1=[0 0;0 1;1 0;1 1]';
A1=sim(net1,P1);
%输出感知器层仿真，并输出仿真结果
P2=ones(3,4);
P2=P2.*A1;
A2=sim(net2,P2)
```



输出仿真结果为：

```
A2 =
     1     1     1     0
```

7.4 径向神经网络仿真分析

7.4.1 径向神经网络应用函数

1．神经网络创建函数

(1) newrb 函数

功能：该函数可以用来设计一个径向基网络。

其调用格式如下：

```
[net,tr] = newrb(P,T,goal,spread,MN,DF)
```

其参数说明如下：

- P：Q 组输入向量组成的 $R\times Q$ 维矩阵。
- T：Q 组目标分类向量组成的 $S\times Q$ 维矩阵。
- goal：均方误差，默认为 0。
- spread：径向基函数的扩展速度，默认为 1。
- MN：神经元的最大数目，默认为 Q。
- DF：两次显示之间所添加的神经元数目，默认为 25。
- net：返回值，一个径向基函数。
- tr：返回值，训练记录。

【例 7-12】 newrb 函数示例。

```
>> P = [1 2 3];
T = [2.0 4.1 5.9];
net = newrb(P,T);
P = 1.5;
Y = sim(net,P)
```

运行程序，输出结果如下：

```
NEWRB, neurons = 0, MSE = 2.54
Y =
    2.6755
```

（2）newrbe 函数

功能：该函数用于设计一个准确的径向基函数。

其调用格式如下：

```
net = newrbe(P,T,spread)
```

各参数含义参见 newrb。

一般来说，newrbe 和 newrb 一样，神经元数目越大，对函数的拟合就越平滑。但是，过多的神经元可能会导致计算困难问题。

【例 7-13】 newrbe 函数示例。

```
>> P = [1 2 3];
T = [2.0 4.1 5.9];
net = newrbe(P,T);
P = 1.5;
Y = sim(net,P)
```

运行程序，输出结果如下：

```
Y =
    2.8054
```

（3）newpnn 函数

功能：该函数可用于创建概率神经网络，概率神经网络是一种适用于分类问题的径向基网络。

其调用格式如下：

```
net = newpnn(P,T,spread)
```

各参数含义参见 newrb。

【例 7-14】 newpnn 函数示例。

```
>> P = [1 2 3 4 5 6 7];
Tc = [1 2 3 2 2 3 1];
T = ind2vec(Tc);
net = newpnn(P,T);
Y = sim(net,P)
Yc = vec2ind(Y)
```

运行程序，输出结果如下：

```
Y =
   (1,1)          1
```



```
     (2,2)          1
     (3,3)          1
     (2,4)          1
     (2,5)          1
     (3,6)          1
     (1,7)          1
Yc =
     1     2     3     2     2     3     1
```



2. 转换函数

（1）ind2vec 函数

功能：该函数用于将数据索引转换为向量组。

其调用格式如下：

```
vec = ind2vec(ind)
```

其参数说明如下：

- ind：数据索引列向量。
- vec：函数返回值，一个稀疏矩阵，每行只有一个 1，矩阵的行数等于数据索引的个数，列数等于数据索引中的最大值。

【例 7-15】 ind2vec 函数示例。

```
>> ind = [1 3 2 3]
vec = ind2vec(ind)
```

运行程序，输出结果如下：

```
ind =
     1     3     2     3
vec =
     (1,1)          1
     (3,2)          1
     (2,3)          1
     (3,4)          1
```

（2）vec2ind 函数

功能：该函数用于将向量转换为数据索引，与 ind2vec 是互逆的。

```
>> vec = [1 0 0 0; 0 0 1 0; 0 1 0 1]
ind = vec2ind(vec)
```

【例 7-16】 vec2ind 函数示例。

```
>> vec = [1 0 0 0; 0 0 1 0; 0 1 0 1]
ind = vec2ind(vec)
```

运行程序，输出结果如下：

```
vec =
```

```
     1     0     0     0
     0     0     1     0
     0     1     0     1
ind =
     1     3     2     3
```

（3）传递函数 radbas

功能：该函数为径向基传递函数。

其调用格式如下：

```
A = radbas(N,FP)
info = radbas(code)
```

其参数说明如下：

- N：输入（列）向量的 $S\times Q$ 维矩阵。
- A：函数返回函数，与 N 一一对应，即 N 中的每个元素通过径向基函数得到 A。
- info = radbas(code)：根据 code 值的不同返回有关函数的不同信息，包括 derive——返回民函数的名称；name——函数名称；output——输出范围；active——返回可用输入范围。

【例 7-17】 radbas 函数示例。

```
>> n = -5:0.1:5;
a = radbas(n);
plot(n,a)
```

运行程序，输出效果如图 7-19 所示。

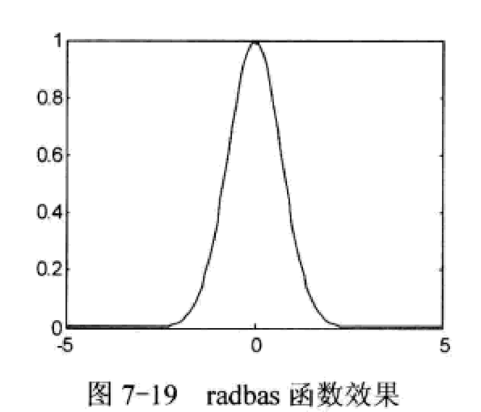


图 7-19 radbas 函数效果

7.4.2 径向神经网络仿真设计分析

【例 7-18】 利用广义回归网络实现函数逼近。

```
%待逼近函数的表达式为
T=cos(4*P)-sin(2*P);
%函数自变量的变化范围是
P=-1:0.2:1;
%利用 newgrnn 函数设计广义回归网络
net=newgrnn(P, T, sp);
```



```
%在广义回归网络中，扩展常数 sp 的选取决定了网络的逼近性能，一般来说，扩展常数和输入
%数据的平均间距相当。本例中将选取两个扩展常数分别建立广义回归网络，并对网络的性能进
%行比较
%本例的 MATLAB 程序代码如下：
>> clear all;
%P 是输入矢量,T 是目标矢量
P=-1:0.2:1;
T=cos(4*P)-sin(2*P);
%画出待逼近函数的图形
plot(P,T,'rp');
%设计两个广义回归网络,对函数进行逼近,第一网络
sp1=0.05;                          %扩展常数 1
net1=newgrnn(P,T,sp1);             %网络 1
%第二个网络
sp2=0.7;                           %扩展常数 2
net2=newgrnn(P,T,sp2);             %网络 2
%利用一组新数据对网络进行测试
P1=-1.1:0.2:1.1;
%对网络系统进行仿真,并画出样本数据图形和网络输出图形
plot(P,T,'r.','markersize',20);
title('第一个网络');
hold on;
Y1=sim(net1,P1);
plot(P1,Y1,'*','markersize',10,'color',[1 0 1]);
%对网络 2 进行仿真,并画出样本数据图形和网络输出图形
figure;
plot(P,T,'r.','markersize',20);
hold on;
Y2=sim(net2,P1);
title('第二个网络');
plot(P1,Y2,'p','markersize',10,'color',[1 0 1]);
```

程序运行结果如图 7-20 所示，图中的红点标注的是样本数据，星号标注的是网络对测试数据的输出。从图中可以看到，由于网络 2 的扩展常数取得过大，因此该网络对数据的细节变化部分不能成功逼近。

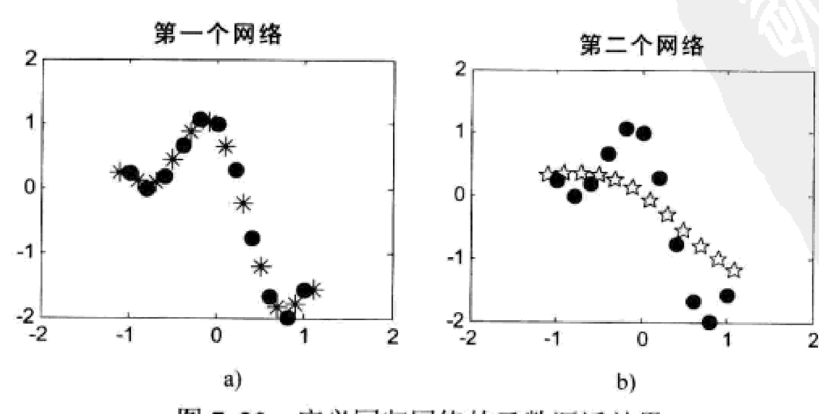


图 7-20 广义回归网络的函数逼近效果

a) 第一个网络 b) 第二个网络

【例 7-19】 已知某系统输出 y 与输入 x 的部分的对应关系见表 7-2。设计一个 RBF 神经网络，完成 $y=f(x)$ 的曲线拟合。

表 7-2 函数 y=f(x)的部分对应关系

x	-1	-0.9	-0.8	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1
y	-0.832	-0.423	-0.024	0.344	1.282	3.456	4.02	3.232	2.102	1.540
x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
y	0.248	1.242	2.344	3.262	2.052	1.684	1.022	2.224	3.022	1.984

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%定义输入向量和目标向量
p=-1:0.1:0.9;
t=[-0.832  -0.423  -0.024  0.344   1.282   3.456   4.02    3.232   2.102   1.540...
    0.248  1.242   2.344   3.262   2.052   1.684   1.022   2.224   3.022   1.984];
%设计径向基网络
t1=clock;    %计时开始
net=newrb(p,t,0.1,0.1,20,5);
datat=etime(clock,t1)
%存储训练好的神经网络
save net720 net;
```

运行程序，输出结果如下：

```
NEWRB, neurons = 0, MSE = 1.73805
NEWRB, neurons = 5, MSE = 0.250454
datat =
    1.6030
```

训练的误差性能曲线如图 7-21 所示。

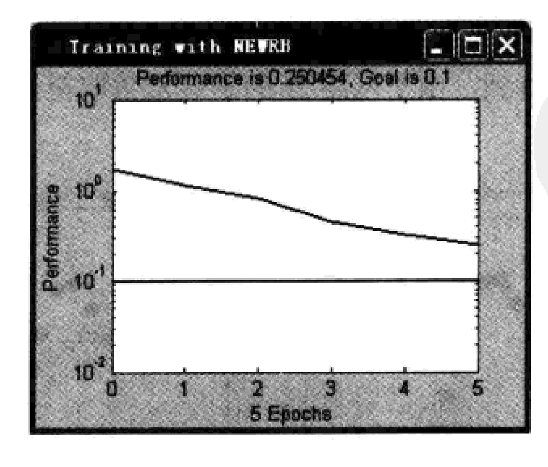


图 7-21 网络设计的误差性能曲线

```
%RBF 神经网络的 MATLAB 仿真程序
>> clear all;
```



```
%绘制训练样本图形
p=-1:0.1:0.9;
t=[-0.832  -0.423  -0.024  0.344   1.282   3.456   4.02    3.232   2.102   1.540...
     0.248  1.242   2.344   3.262   2.052   1.684   1.022   2.224   3.022   1.984];
hold on;
plot(p,t,'mp');
%网络仿真
load net720 net;
i=-1:0.05;0.9;
r=sim(net,i);
plot(i,r,'-.');
%绘制函数拟合曲线
plot(i,r);
hold off
```

运行程序，输出效果如图 7-22 所示，虚线为得到的拟合曲线，“星号”为训练样本。

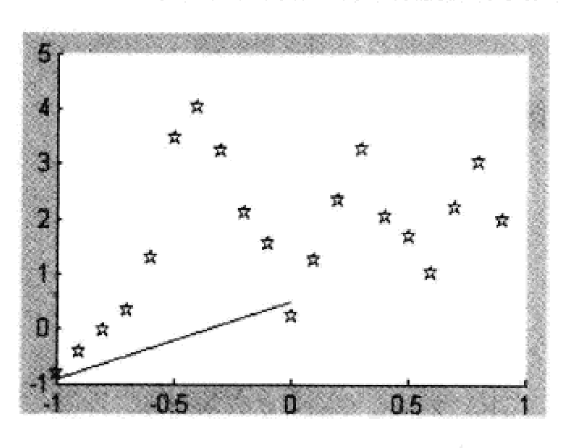


图 7-22 仿真结果

7.5 BP 神经网络仿真分析

7.5.1 BP 神经网络应用函数

MATLAB R2009 神经网络工具箱中包含了许多用于 BP 网络分析与设计的函数，本节分别说明这些函数的功能、调用格式和注意事项等。

1. BP 网络的创建函数

(1) newcf 函数

功能：该函数用于创建级联前向 BP 网络。

其调用格式如下：

```
net=newcf
net = newcf(P,T,[S1 S2…S(N-l)],{TF1 TF2…TFN}, BTF,BLF,PF,IPF,OPF,DDF)
```

其参数说明如下：

- net=newcf：用于在对话框中创建一个 BP 网络。

● P：由每组输入（共有 R 组输入）元素的最大值和最小值组成的 $R\times2$ 维的矩阵。
● T：Q 组目标分类向量组成的 $S\times Q$ 维矩阵。
● Si：第 i 层的长度，共计 NI 层。
● TFi：第 i 层的传递函数，默认为 'tansig'。
● BTF：BP 网络的训练函数，默认为 'trainlm'。
● BLF：权值和阈值 BP 学习算法，默认为 'learngdm'。
● PF：网络的性能函数，默认为 'mse'。

【例 7-20】 newcf 函数示例。

```
>> P = [0 1 2 3 4 5 6 7 8 9 10];
T = [0 1 2 3 4 3 2 1 2 3 4];
net = newcf(P,T,5);
net.trainParam.epochs = 50;
net = train(net,P,T);
Y = sim(net,P);
plot(P,T,P,Y,'o')
```

运行程序，输出效果如图 7-23 所示。

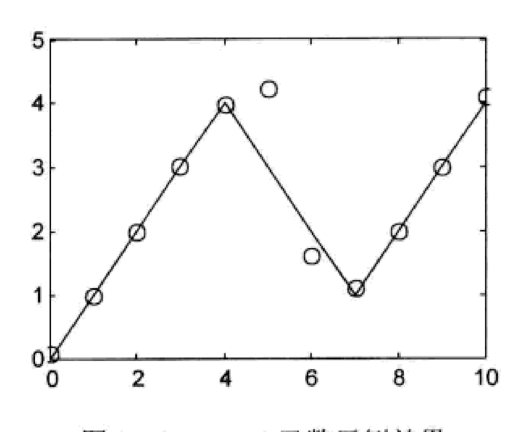


图 7-23 newcf 函数示例效果

（2）newff 函数

功能：该函数用于创建一个 BP 网络。

其调用格式如下：

```
net=newff
net = newff(P,T,[S1 S2…S(N-l)],{TF1 TF2…TFNl}, BTF,BLF,PF,IPF,OPF,DDF)
net=newff：用于在对话框中创建一个 BP 网络。
```

其参数说明参考 newcf 函数。

【例 7-21】 newff 函数示例。

```
>> P = [0 1 2 3 4 5 6 7 8 9 10];
T = [0 1 2 3 4 3 2 1 2 3 4];
net = newff(P,T,5);
net.trainParam.epochs = 50;
net = train(net,P,T);
```

```
Y = sim(net,P);
plot(P,T,P,Y,'o')
```

运行程序，输出效果如图 7-24 所示。

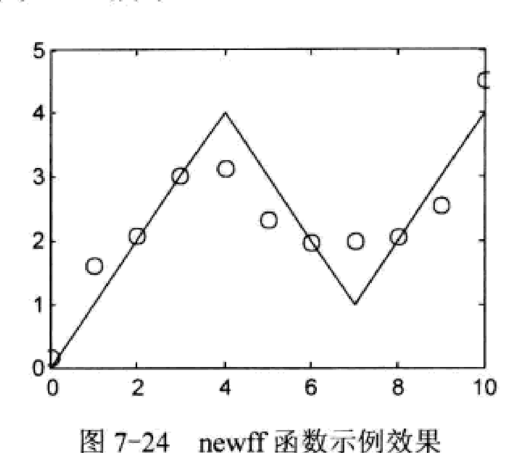




图 7-24　newff 函数示例效果

（3）newfftd 函数

功能：该函数用于创建一个存在输入延迟的前向网络。

其调用格式如下：

```
net = newfftd(P,T,ID,[S1 S2…S(N-l)],{TF1 TF2…TFNl}, BTF,BLF,PF,IPF,OPF,DDF)
net=newfftd：用于在对话框中创建一个 BP 网络。
```

其参数说明参考 newcf 函数。

2. 神经元上的传递函数

传递函数是 BP 网络的重要组成部分。传递函数又称为激活函数，必须是连续可微的。BP 网络经常采用 S 型的对数或正切函数以及线性函数。

（1）logsig 函数

功能：该传递函数为 S 型的对数函数。

其调用格式如下：

```
A = logsig(N,FP)
info = logsig(code)
```

其参数说明如下：

- N：Q 个 S 维的输入列向量。
- FP：结构函数的参数。
- A：函数返回值，位于区间（0,1）中。
- info = logsig(code)：依据 code 值的不同返回不同的信息，包括 derive——返回微分函数的名称；name——返回函数全称；output——返回输出值域；active——返回有效的输入区间。

【例 7-22】 logsig 函数示例。

```
>> n = -5:0.1:5;
a = logsig(n);
plot(n,a)
```

运行程序，输出效果如图 7-25 所示。

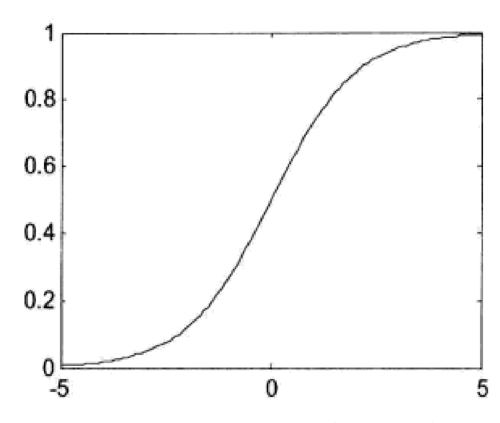


图 7-25　logsig 函数输出效果

（2）tansig 函数

功能：该函数为双曲正切 S 型函数。

其调用格式如下：

```
A = tansig(N,FP)
info = tansig(code)
```

其参数说明如下：

- N：Q 个 S 维的输入列向量。
- FP：结构函数的参数。
- info = tansig(code)：的含义参见 logsig(code)。

【例 7-23】 tansig 函数示例。

```
>> n = -5:0.1:5;
a = tansig(n);
plot(n,a)
```

运行程序，输出效果如图 7-26 和图 7-27 所示。

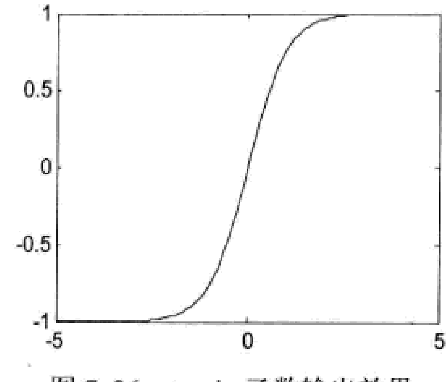


图 7-26　tansig 函数输出效果

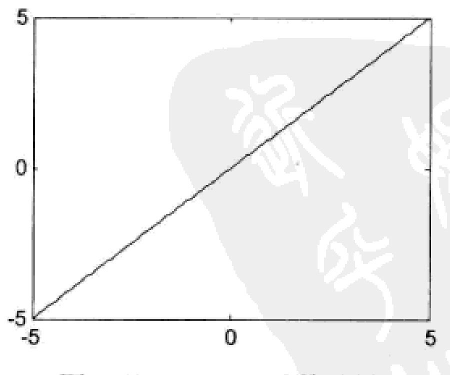


图 7-27　purelin 函数效果

（3）purelin 函数

功能：该函数为线性传递函数。

其调用格式如下：

```
A = purelin(N,FP)
info = purelin(code)
```



其参数说明如下：

- N：Q 个 S 维的输入列向量。
- FP：结构函数的参数。
- A：函数返回值，$A=N$。
- info = purelin(code)：的含义参见 logsig(code)。

【例 7-23】 purelin 函数示例。

```
>> n = -5:0.1:5;
a = purelin(n);
plot(n,a)
```

3. BP 网络的学习函数

（1）learngd 函数

功能：该函数为梯度下降权值/阈值学习函数，它通过神经元的输入和误差，以及权值和阈值的学习速率，来计算权值或阈值的变化率。

其调用格式如下：

```
[dW,LS] = learngd(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
[db,LS] = learngd(b,ones(1,Q),Z,N,A,T,E,gW,gA,D,LP,LS)
info = learngd(code)
```

其参数说明如下：

- W：$S \times R$ 维的权值矩阵。
- P：Q 组 R 维的输入向量。
- Z：Q 组 S 维的加权输入向量。
- N：Q 组 S 维的输入向量。
- A：Q 组 S 维的输出向量。
- T：Q 组 S 维的层目标向量。
- E：Q 组 S 维的层误差向量。
- gW：与性能相关的 $S \times R$ 维梯度。
- gA：与性能相关的 $S \times R$ 维输出梯度。
- D：$S \times S$ 维的神经元距离矩阵。
- LP：学习参数，可通过该参数设置学习速度，设置格式如 LP.lr=0.01。
- LS：学习状态，初始状态下为空。
- b：S 维的阈值向量。
- ones(1,Q)：产生个 Q 维的输入向量。
- dW：$S \times R$ 维的权值或阈值变化率矩阵。
- ls：新的学习状态。
- learngd(code)：根据不同的 code 值返回有关函数的不同信息，包括 pnames——返回

设置的学习参数；pdefaults——返回默认的学习参数；needg——如果函数使用 gW 或 gA，则返回 1。

【例 7-24】 learngd 函数示例。

```
>> gW = rand(3,2);
lp.lr = 0.5;
dW = learngd([],[],[],[],[],[],[],gW,[],[],lp,[])
```

运行程序，输出结果如下：

```
dW =
     0.4074     0.4567
     0.4529     0.3162
     0.0635     0.0488
```

（2）learngdm 函数

功能：该函数为梯度下降动量学习函数，它利用神经元的输入和误差、权值或阈值的学习速度和动量常数，来计算权值或阈值的变化率。

其调用格式如下：

```
[dW,LS] = learngdm(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
[db,LS] = learngdm(b,ones(1,Q),Z,N,A,T,E,gW,gA,D,LP,LS)
info = learngdm(code)
```

其各参数的含义参见 learngd。

4．网络训练函数

（1）trainbfg 函数

功能：该函数为 BFGS 准牛顿 BP 算法函数。除了 BP 网络外，该函数也可以训练任意形式的神经网络，只要它的传递函数对于权值和输入存在导数即可。

其调用格式如下：

```
[net,TR] = trainbfg(NET,Tr,trainV,valV,testV)
info = trainbfg('info')
```

其参数说明如下：

- NET：待训练的神经网络。
- Tr：有延迟的输入向量。
- trainV：层次目标向量。
- valV：确认向量结构或者为空。
- testV：检验向量结构或者为空。
- net：训练后的神经网络。
- TR：每步训练的有关信息记录，包括 TR.epoch——时刻点；TR.perf——训练性能；TR.vperf——确认性能；TR.tperf——检验性能。
- trainbfg('info')：根据不同的 code 值返回不同的有关 trainbfg 的信息，包括 pnames——返回设定的训练参数；pdefaults——返回默认的训练参数，其参数名

称与属性见表 7-3。

表 7-3 BP 网络训练参数



参数名称	默认 1 值	属 性
net.trainParam.epochs	100	训练次数。100 为训练次数的最大值
net.trainParam.show	25	两次显示之间的训练步数（无显示时设为 NaN）
net.trainParam.goal	0	训练目标
net.trainParam.time	inf	训练时间，inf 表示训练时间不限
net.trainParam.min_grad	1e-6	最小性能梯度
net.trainParam.max_fail	5	最大确认失败次数
net.trainParam.searchFcn	'srchcha'	所用的线性搜索路径



（2）traingdm 函数

功能：该函数为梯度下降动量 BP 算法函数。

其调用格式如下：

```
[net,TR] = traingdm(net,TR,trainV,valV,testV)
info = traingdm('info')
```

其各参数的含义与设置格式及适用范围参见 trainbfg。

（3）traingd 函数

功能：该函数为梯度下降 BP 算法函数。

其调用格式如下：

```
[net,TR] = traingd(net,TR,trainV,valV,testV)
info = traingd('info')
```

其各参数的含义与设置格式及适用范围参见 trainbfg。

另外，MATLAB R2009 的神经网络工具箱中还有一系列训练函数可用于对 BP 网络的训练，在此不再一一介绍，使用时读者可以比照函数 trainbfg 的调用格式进行学习。

5．性能函数

（1）mse 函数

功能：该函数为均方误差性能函数。

其调用格式如下：

```
perf = mse(E,Y,X,FP)
info = mse(code)
```

其各参数含义参见 mae。

【例 7-25】 mse 函数示例。

```
>> net = newff([-10 10],[4 1],{'tansig','purelin'});
p = [-10 -5 0 5 10];
t = [0 0 1 1 1];
y = sim(net,p);
```

```
e = t-y;
perf = mse(e)
```

运行程序，输出结果如下：

```
perf =
     1.5906
```

（2）msereg 函数

功能：该函数也是性能函数，它通过两个因子的加权和来评价网络的性能，这两个因子分别是均方误差、均方权值和阈值。

其调用格式如下：

```
perf = msereg(E,Y,X,FP)
info = msereg(code)
```

其各参数含义参见 mae。

【例 7-26】 msereg 函数示例。

```
>> p = [-2 -1 0 1 2];
t = [0 1 1 1 0];
y = sim(net,p);
e = t-y;
net.performParam.ratio = 20/(20+1);
perf = msereg(e,net)
```

运行程序，输出结果如下：

```
perf =
     1.8840
```

6. 显示函数

（1）plotes 函数

功能：该函数用于绘制一个单独神经元的误差曲线。

其调用格式如下：

```
plotes(wv, bv, es, v)
```

其参数说明如下：

- wv：权值的 N 维行向量。
- bv：M 维的阈值行向量。
- es：误差向量组成的 $M\times N$ 维矩阵。
- v：视角，默认为[-37.5, 30]。

【例 7-27】 plotes 函数示例。

```
>> p = [3 2];
t = [0.4 0.8];
wv = -4:0.4:4; bv = wv;
```

```
ES = errsurf(p,t,wv,bv,'logsig');
plotes(wv,bv,ES,[60 30])
```

运行程序，输出效果如图 7-28 所示。

（2）plotperf 函数

功能：该函数用于绘制网络的性能。

其调用格式如下：

```
plotper(tr,goal,name,epoch)
```

其参数说明如下：

- tr：网络训练记录。
- goal：性能目标，默认为 NaN。
- name：训练函数名称，默认为空。
- epoch：训练步数，默认为训练记录的长度。

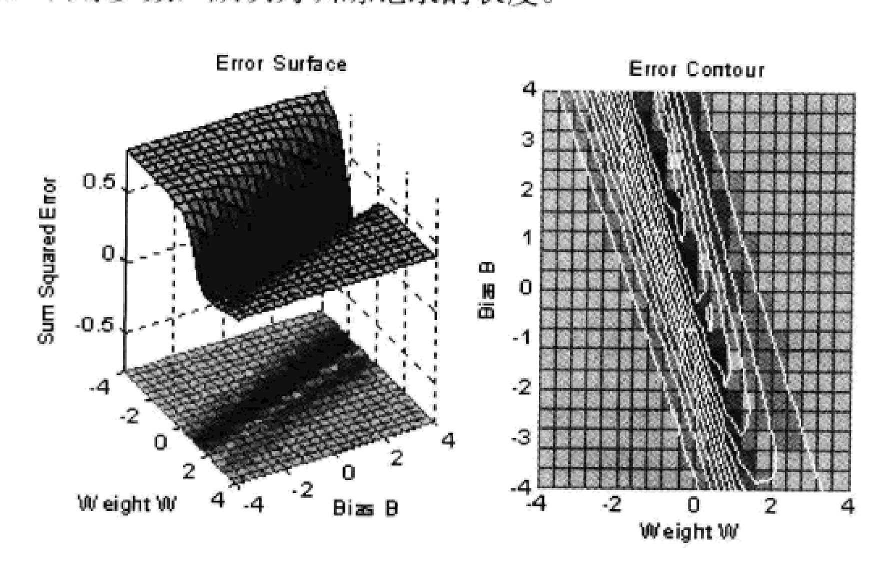


图 7-28 plotes 函数效果图



（3）plotep 函数

功能：该函数用于绘制权值和阈值在误差曲面上的位置。

其调用格式如下：

```
h = plotep(W,B,E)
h = plotep(W,B,E,H)
```

其参数说明如下：

- W：当前权值。
- B：当前阈值。
- E：当前单输入神经元的误差。
- h：权值和阈值在上一时刻的位置信息向量。
- H：当前的权值和阈值位置信息向量。

（4）errsurf 函数

功能：此函数用于计算单个神经元的误差曲面。神经元的误差曲面是由权值和阈值的行向量确定的。

其调用格式如下：

```
errsurf(P,T,WV,BV,F)
```

其参数说明如下：

- P：输入行向量。
- T：目标行向量。
- WV：权值列向量。
- BV：阈值列向量。
- F：传递函数的名称。

【例 7-28】 plotep 函数示例。

```
>> X=[3 2];T=[0.4 0.8];
W=-4:0.4:4;b=W;
es=errsurf(X,T,W,b,'logsig');
plotes(W,b,es,[60 30]);
W=-2;b=0;
e=sumsqr(T-simuff(X,W,b,'logsig'));
plotep(W,b,e);
```

运行程序，输出效果如图 7-29 所示。

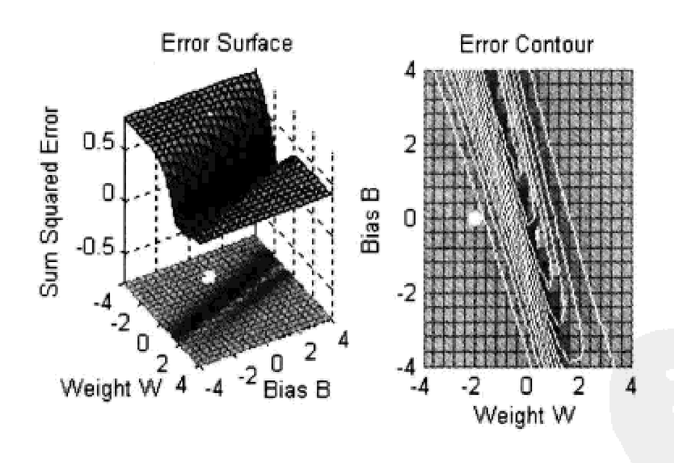


图 7-29 权值和偏值在误差曲面图上的位置

7.5.2 BP 神经网络仿真设计分析

【例 7-29】 利用两层 BP 神经网络训练加权系数。两组 3 输入为[1 2; -1 1; 1 3]，希望的输出均为[1, 1]。隐含层的激活函数取 S 型传输函数，输出层的激活函数取线性传输函数。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%两层 BP 算法的第一阶段：学习期(训练加权系数 Wki, Wij)
```



```
%初始化
lr=0.05;                              %lr 为学习速率
err_goal=0.001;                       %err_goal 为期望误差最小值
max_epoch=10000;                      %训练的最大次数
x=[1 2;-1 1;1 3]; T=[1 1;1 1];        %提供两组 3 输入 2 输出的训练集和目标值
%初始化 Wki, Wij(M 为输入节点 j 的数量,q 为隐含层节点 i 的数量,L 为输出节点 k 的数%量)
[M,N]=size(x);
q=10;
[L,N]=size(T);
Wij=rand(q,M);                        %随机给定输入层与隐含层间的权值
Wki=rand(L,q);                        %随机给定隐含层与输出层间的权值
b1=zeros(q,1);
b2=zeros(L,1);
for epoch=1:max_epoch
    oi=tansig(Wij*x,b1);              %计算网络隐含层的各神经元输出
    ok=purelin(Wki*oi,b2);            %计算网络输出层的各神经元输出
    e=T-ok;                           %计算网络误差
    deltak=deltalin(ok,e);            %计算输出层的 delta
    deltai=deltatan(oi,deltak,Wki);   %计算隐含层的 delta
    %调整输出层加权系数
    [dwki,db2]=learnbp(oi,deltak,lr);
    Wki=Wki+dwki;
    b2=b2+db2;
    [dwij,db1]=learnbp(x,deltai,lr);
    Wij=Wij+dwij;
    b1=b1+db1;
    %计算网络权值修正后的误差平方和
    sse=sumsqr(T-purelin(Wki*tansig(Wij*x,b1),b2));
    if(sse<err_goal)
        break;
    end
end
%BP 算法的第二阶段：工作期(根据训练好的 Wki,Wij 和给定的输入计算输出)
x1=x;
oi=tansig(Wij*x1,b1);                 %计算网络隐含层的各神经元输出
ok=purelin(Wki*oi,b2)                 %计算网络输出层的各神经元输出
```

运行程序，输出结果如下：

```
ok =
    1.0036    0.9980
    0.9759    1.0190
```

【例 7-30】 已知某系统输出 y 与输入 x 的部分的对应关系见表 7-2。设计一个 BP 神经网络，完成 $y=f(x)$ 的曲线拟合，并与例 7-19 的显示效果进行对比。

```
%以隐层节点数为 15 的单输入和单输出两层 BP 网络来实现其曲线拟合
```

```
>> clear all;
%定义输入向量和目标向量
p=-1:0.1:0.9;
t=[-0.832  -0.423 -0.024 0.344  1.282  3.456  4.02   3.232  2.102  1.540...
    0.248 1.242  2.344  3.262  2.052  1.684  1.022  2.224  3.022  1.984];
%设计径向基网络
net=newff([-1 1],[15 1],{'tansig','purelin'},'traingdx','learngdm');
net.trainParam.epochs=2500;
net.trainParam.goal=0.001;
net.trainParam.show=10;
net.trainParam.lr=0.05;
net=train(net,p,t)
%存储训练好的神经网络
save net730 net;
```

运行程序，输出结果如下：

```
TRAINGDX, Epoch 0/2500, MSE 13.1049/0.001, Gradient 18.0373/1e-006
TRAINGDX, Epoch 10/2500, MSE 1.10623/0.001, Gradient 0.955673/1e-006
TRAINGDX, Epoch 20/2500, MSE 0.851304/0.001, Gradient 0.755135/1e-006
……
TRAINGDX, Epoch 2500/2500, MSE 0.019539/0.001, Gradient 0.00939566/1e-006
TRAINGDX, Maximum epoch reached, performance goal was not met.
```

训练的误差性能曲线如图 7-30 所示。

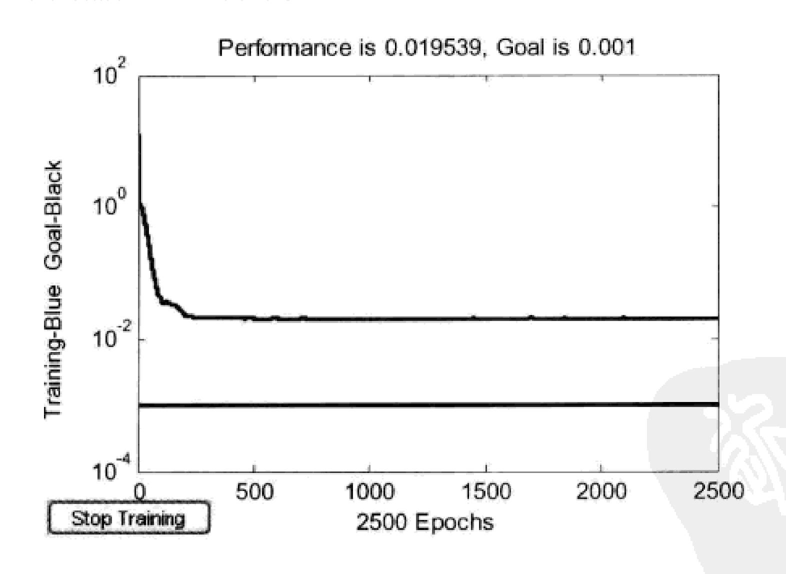


图 7-30　训练的误差性能曲线

```
%BP 网络仿真的 MATLAB 程序
>> clear all;
%绘制训练样本图形
p=-1:0.1:0.9;
t=[-0.832  -0.423 -0.024 0.344  1.282  3.456  4.02   3.232  2.102  1.540...
    0.248 1.242  2.344  3.262  2.052  1.684  1.022  2.224  3.022  1.984];
```

```
hold on;
plot(p,t,'mp');
%网络仿真
load net730 net;
p=-1:0.01;0.9;
r=sim(net,p);
plot(p,r,'-.');
%绘制函数拟合曲线
plot(p,r);
hold off
```



曲线拟合结果如图 7-31 所示。实线为得到的拟合曲线；“星号”为训练样本。从结果可以看出，可以对训练样本进行很好的拟合，但拟合曲线欠光滑，出现了“过适配”现象。

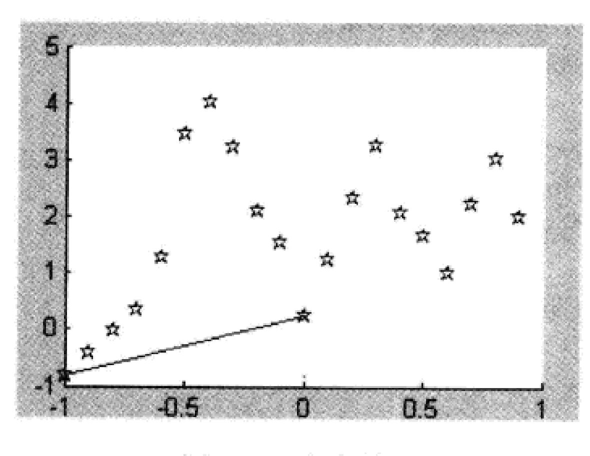


图 7-31 仿真结果

【例 7-31】 采用贝叶斯正则化算法提高 BP 网络的推广能力。在本例中，我们将采用两种训练方法，即 L-M 优化算法（Trainlm）和贝叶斯正则化算法（Trainbr），用以训练 BP 网络，使其能够拟合某一附加有白噪声的正弦样本数据。其中，样本数据可以采用如下 MATLAB 语句生成：

```
输入矢量：P=[-1:0.05:1]
目标矢量：randn('seed', 78341223);
          T=sin(2*pi*P)+0.1*randn(size(P));
```

其实现的 MATLAB 程序代码如下：

```
>> clear all;
%定义训练样本矢量
P=[-1:0.05:1];
% T 为目标矢量
randn('seed', 78341223);
T=sin(2*pi*P)+0.1*randn(size(P));
%绘制样本数据点
plot(P,T,'mp');
hold on;
plot(P,sin(2*pi*P),'-.');     %绘制不含噪声的正弦曲线
```

```
%创建一个新的前向神经网络
net=newff(minmax(P),[20,1],{'tansig','purelin'});
disp('1.L-M 优化算法 TRAINLM');
disp('2.贝叶斯正则化算法 TRAINBR');
choice=input('请选择训练算法(1,2)');
if(choice==1)
    %采用 L-M 优化算法 TRAINLM
    net.trainFcn='trainlm';
    %设置训练参数
    net.trainParam.epochs=500;
    net.trainParam.goal=1e-6;
    net=init(net);  %重新初始化
elseif (choice==2)
    %采用贝叶斯正则化算法 TRAINBR
    net.trainFcn='trainbr';
    %设置训练参数
    net.trainParam.epochs=500;
    randn('seed',192736547);
    net=init(net);   %重新初始化
end
% 调用相应算法训练 BP 网络
[net,tr]=train(net,P,T);
%对 BP 网络进行仿真
a=sim(net,P);
%计算仿真误差
e=T-a;
ms=mse(e);
%绘制匹配结果曲线
plot(P,a,P,T,'mp',P,sin(2*pi*P),'-.');
```

通过采用两种不同的训练算法，我们可以得到如图 7-32 和图 7-33 所示的两种拟合结果。图中的实线表示拟合曲线，虚线代表不含白噪声的正弦曲线，“星号”点为含有白噪声的正弦样本数据点。显然，经 trainlm 函数训练后的神经曲线对样本数据点实现了“过度匹配”，而经 trainbr 函数训练的神经网络对噪声不敏感，具有较好的推广能力。

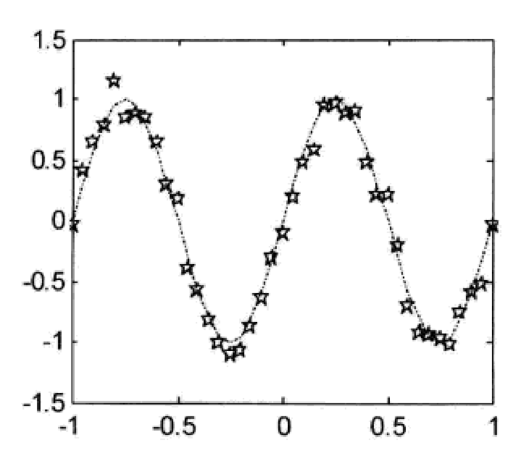


图 7-32 trainlm 训练后的拟合结果

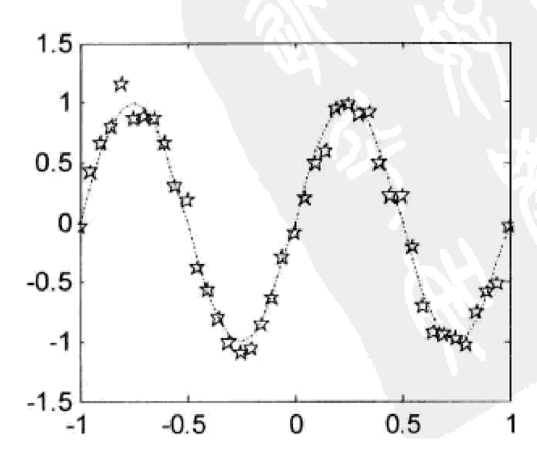


图 7-33 trainbr 训练后的拟合结果

值得指出的是，在利用 trainbr 函数训练 BP 网络时，若训练结果收敛，通常会给出提示信息“Maximun MU reached”。此外，用户还可以根据 sse 和 ssw 的大小变化情况来判断训练是否收敛：当 sse 和 ssw 的值在经过若干步迭代后处于恒值时，则通常说明网络训练收敛，此时可以停止训练。图 7-34 给出了本例中利用 trainlm 函数训练 BP 网络的误差变化曲线。可见，当训练迭代至 256 步时，网络训练收敛，此时 sse 和 ssw 均为恒值，当前有效网络的参数（有效权值和阈值）个数为 11.7973。



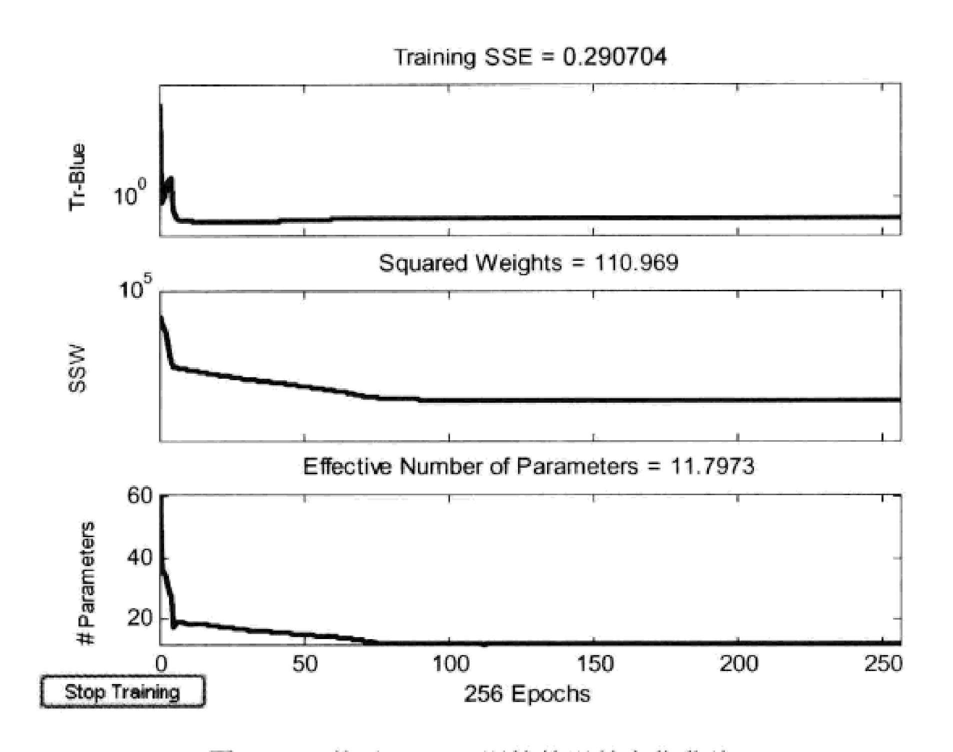


图 7-34 基于 trainlm 训练的误差变化曲线

7.6 自组织神经网络的函数及其 MATLAB 实现

1. 神经网络创建函数

（1）newsom 函数

功能：该函数用于创建一个自组织特征映射。

其调用格式如下：

```
net = newsom
net = newsom(PR,[D1,D2,…],TFCN,DFCN,OLR,OSTEPS,TLR,TND)
```

其参数说明如下：

- net=newsom：表示在对话框中创建一个新的网络。
- PR：R 个输入元素的最大值和最小值的设定值，$R\times 2$ 维矩阵。
- Di：第 i 层的维数，默认为[5　8]。
- TFCN：拓扑函数，默认为“hextop”。
- DFCN：距离函数，默认为“linkdist”。

● OLR：分类阶段学习速率，默认为 0.9。
● OSTEPS,：分类阶段的步长，默认为 1000。
● TLR：调谐阶段的学习速率，默认为 0.02。
● TNS：调谐阶段的领域距离，默认为 1。

函数返回一个自组织特征映射。

【例 7-32】 newsom 函数示例。

```
>> P = [rand(1,400)*2; rand(1,400)];
net = newsom([0 2; 0 1],[3 5]);
figure(1);
plotsom(net.layers{1}.positions);
net = train(net,P);
figure(2);
plot(P(1,:),P(2,:),'.g','markersize',20)
hold on
plotsom(net.iw{1,1},net.layers{1}.distances)
hold off
```

运行程序，效果如图 7-35 所示。

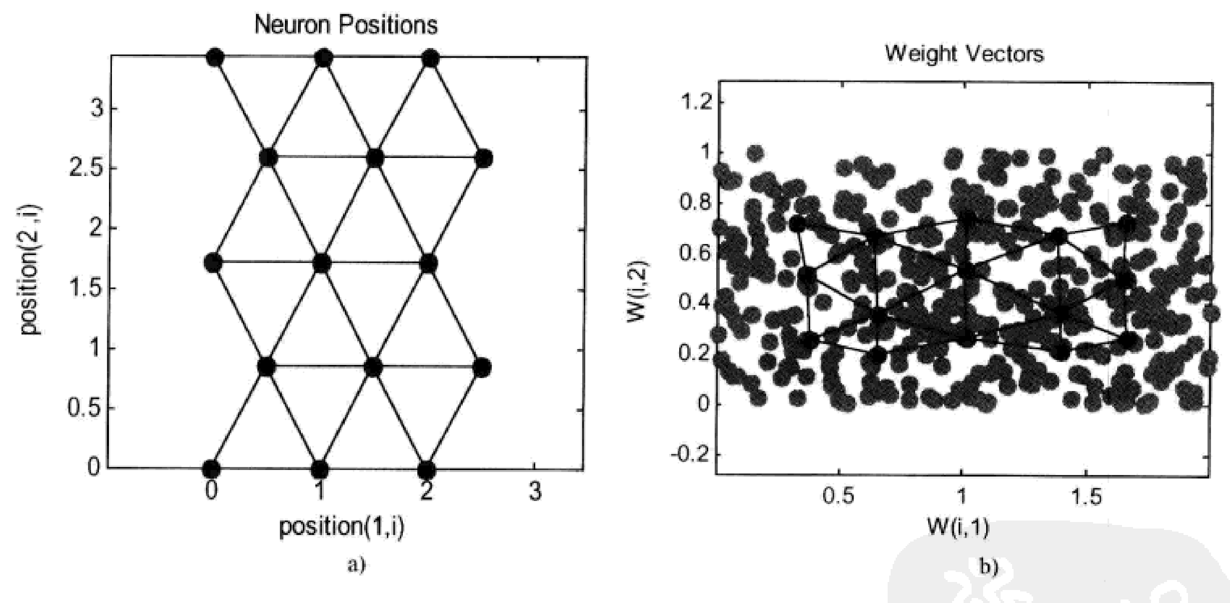


图 7-35 newsom 函数创建的特征映射

a) 初始权值的分布 b) 样本点与训练后权值分布

（2）newc 函数

功能：该函数用于创建一个竞争层。

其调用格式如下：

```
net = newc
net = newc(PR,S,KLR,CLR)
```

其参数说明如下：



- net = newc：表示在对话框中创建一个新的
- PR：R 个输入元素的最大值和最小值的设定值，$R\times 2$ 维矩阵。
- S：神经元的数目。
- KLR：Kohonen 学习速度，默认为 0.01。
- CLR：Conscience 学习速度，默认为 0.001。
- net：函数返回值，一个新的竞争层。

【例 7-33】 利用 newc 函数建立一个输入向量分布在一个二维空间、其变化范围分别为[0 1]和[0 1]、用来区分 5 种模式的基本竞争型神经网络，并对其进行训练和仿真。

其实现的 MATLAB 程序代码如下：

```
>> %产生具有 5 类样本类别的样本点,并在图中绘制出
C=[0 1;0 1];cl=5;
points=10;std=0.05;
x=nngenc(C,cl,points,std);
plot(x(1,:),x(2,:),'mp');
xlabel('x(1)');ylabel('x(2)');
% 建立神经元为 5 的一个自组织竞争型神经网络,并求出初始权值
net=newc([0 1;0 1],5,0.1);
w=net.iw{1};                    %初始权值
%训练神经网络,并设置最大训练步数为 7
net.trainParam.epochs=7;        %最大训练步数
net=init(net);
net=train(net,x);
w1=net.iw{1};                   %训练后的权值
plot(x(1,:),x(2,:),'mp');
xlabel('x(1)');ylabel('x(2)');
hold on;
plot(w1(:,1),w1(:,2),'ob');
hold off;
%对于训练好的网络进行测试与使用
x1=[0.6;0.8];
y=sim(net,x1)
```

运行程序，输出结果如下：

```
y =
   (4,1)          1
```

同时，其输出效果如图 7-36 所示。

从图 7-36 可见，网络经过训练以后，权值得到了调整，调整后的权值分布在各个类的中心位置上。对于需要分类的模式向量[0.6; 0.8]，将其输入到训练好的网络中，网络就可以对其分类。分类结果指出了第（4，1）个神经元发生了响应，它反映了这个输入所属的类别。

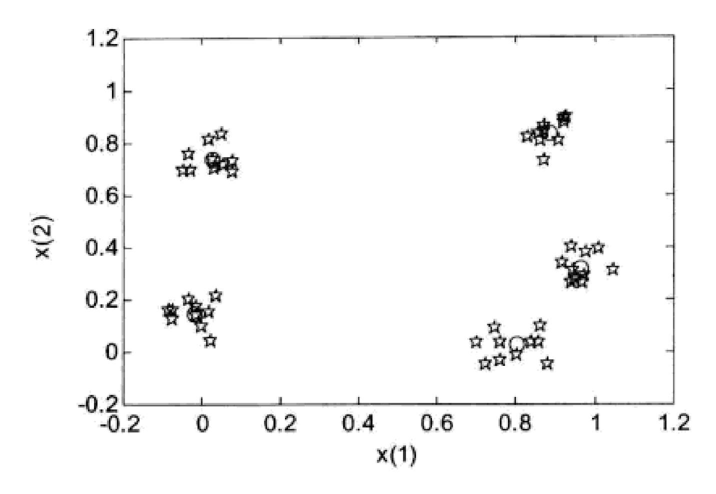


图 7-36　训练后网络权值的分布

（3）newlvq 函数

功能：该函数用于创建一个学习向量量化 LVQ 网络。

其调用格式如下：

```
net = newlvq
net = newlvq(PR,S1,PC,LR,LF)
```

其参数说明如下：

- net = newlvq：用于在对话框中创建一个学习向量量化 LVQ 网络。
- PR：$R\times2$ 的矩阵，指定了输入向量中元素的最大值和最小值。
- S1：竞争层神经元的数目。
- PC：分类的百分比。
- LR：学习速率，默认为 0.01。
- LF：学习函数，默认为 learnlv1。

【例 7-34】 newlvq 函数示例。

```
>> P = [-3 -2 -2   0   0   0   0 +2 +2 +3; ...
0 +1 -1 +2 +1 -1 -2 +! -1   0];
Tc = [1 1 1 2 2 2 2 1 1 1];
T = ind2vec(Tc);
net = newlvq(minmax(P),4,[.6 .4]);
net = train(net,P,T);
Y = sim(net,P);
Yc = vec2ind(Y)
```

运行程序，输出结果如下：

```
Yc =
     1     1     1     2     2     2     2     1     1     1
```

2. 传递函数

（1）compet 函数

功能：该函数为传递函数。

其调用格式如下：

```
A = compet(N)
info = compet(code)
```

参数说明如下：

- N：输入（列）向量的 $S \times Q$ 维矩阵。
- A：函数返回值，输出向量矩阵，每一列向量只有一个 1，位于输入向量最大的位置。
- info = compet(code)：根据 code 值的不同返回有关函数的不同信息，包括 derive——返回导函数名称；name——函数名称；output——输出范围；active——动态输入范围。

【例 7-35】 compet 函数示例。

```
>> n = [0; 1; -0.5; 0.5];
a = compet(n);
subplot(2,1,1), bar(n), ylabel('n')
subplot(2,1,2), bar(a), ylabel('a')
```

运行程序，输出效果如图 7-37 所示。

（2）softmax 函数

功能：该函数为软最大传递函数。

其调用格式如下：

```
A = softmax(N)
info = softmax(code)
```

其参数含义与 compet。与 compet 不同的是，参数 A 为函数返回向量，各元素在区间[0, 1]，且向量结构与 N 一致。

【例 7-36】 softmax 函数示例。

```
>> n = [0; 1; -0.5; 0.5];
a = softmax(n);
subplot(2,1,1), bar(n), ylabel('n')
subplot(2,1,2), bar(a), ylabel('a')
```

运行程序，输出效果如图 7-38 所示。

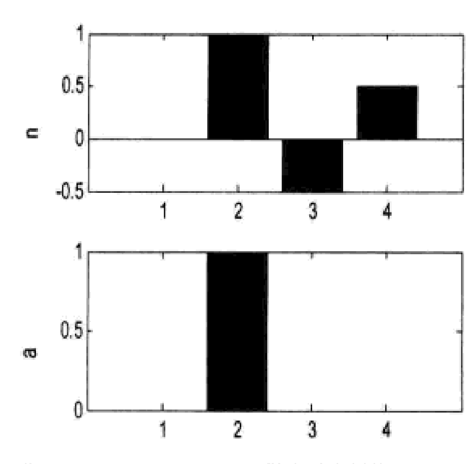


图 7-37 compet 函数绘图效果

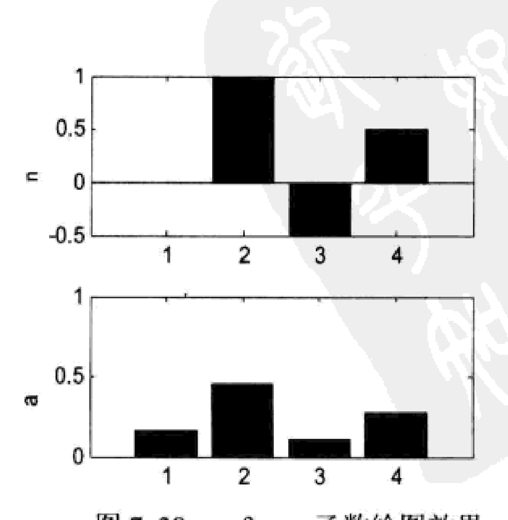


图 7-38 softmax 函数绘图效果



3. 距离函数

（1）boxdist 函数

功能：该函数为 Box 距离函数，在给定神经网络某层的神经元的位置后，可利用该函数计算神经元之间的距离，该函数通常用于结构函数的 gridtop 的神经网络层。

其调用格式如下：

```
d = boxdist(pos);
```

其参数说明如下：

- ps：神经元位置的 $N \times S$ 维矩阵。
- d：函数返回值，神经元距离的 $S \times S$ 维矩阵。

函数的运算原理为 $d(i,j) = \max\|p_i - p_j\|$。其中，$d(i,j)$ 表示距离矩阵中的元素；p_i 表示位置矩阵的第 i 列向量。

【例 7-37】 boxdist 函数示例。

```
>> pos = rand(3,6);
d = boxdist(pos)
```

运行程序，输出结果如下：

```
d =
         0    0.9194    0.8094    0.6144    0.7802    0.6468
    0.9194         0    0.1171    0.6933    0.2195    0.5760
    0.8094    0.1171         0    0.7654    0.3366    0.6481
    0.6144    0.6933    0.7654         0    0.5766    0.1173
    0.7802    0.2195    0.3366    0.5766         0    0.4856
    0.6468    0.5760    0.6481    0.1173    0.4856         0
```

（2）dist 函数

功能：该函数的欧氏距离权函数，通过对输入进行加权得到加权后的输入。

其调用格式如下：

```
Z = dist(W,P)
df = dist('deriv')
D = dist(pos)
```

其参数说明如下：

- W：$S \times R$ 维的权值矩阵。
- P：Q 组输入（列）向量的 $R \times Q$ 维矩阵。
- Z：$S \times Q$ 维的距离矩阵。
- pos：神经元位置的 $N \times S$ 维矩阵。
- D：$S \times S$ 维的距离矩阵。
- df = dist('deriv')：返回值为空，因为该函数不存在导函数。

函数的运算规则为 $D = sqrt(sum((x-y)^2))$，其中 x 和 y 分别为列向量。

```
>> W = rand(4,3);
P = rand(3,1);
Z = dist(W,P)
```



运行程序，输结果如下：

```
Z =
    0.9389
    0.9494
    0.9476
    0.6181
```

（3）linkdist 函数

功能：该函数为连接距离函数，在给定神经元的位置后，该函数可用于计算神经元之间的距离。

其调用格式如下：

```
d = linkdist(pos)
```

其参数说明如下：

- pos：N×S 维的神经元位置矩阵。
- d：S×S 维的距离矩阵。

（4）mandist 函数

功能：该函数为 Manhattan 距离权函数。

其调用格式如下：

```
Z = mandist(W,P)
df = mandist('deriv')
D = mandist(pos);
```

其参数说明参见 dist。

4．学习函数

（1）learnk 函数

功能：该函数为 Kohonen 权值学习函数。

其调用格式如下：

```
[dW,LS] = learnk(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
info = learnk(code)
```

其参数说明如下：

- W：$S\times R$ 维的权值矩阵（或 S 维的阈值向量）。
- P：$R\times Q$ 维的输入向量矩阵[也可用 ones(1,Q)产生]。
- Z：$S\times Q$ 维的权值输入向量矩阵。
- N：$S\times Q$ 维网络输入向量矩阵。
- A：$S\times Q$ 维输出向量矩阵。

- T：$S \times Q$ 层目标向量矩阵。
- E：$S \times Q$ 层误差向量矩阵。
- gW：$S \times Q$ 性能梯度矩阵。
- gA：$S \times Q$ 输出性能梯度矩阵。
- D：$S \times S$ 神经元距离矩阵。
- LP：学习参数，若无则为空。
- LS：学习状态，初始化为空。
- dW：$S \times R$ 维的权值（阈值）变化矩阵。
- LS：新的学习状态。
- info=learnk(code)：根据不同的 code 值返回不同的相关信息，包括 pnames——返回学习参数的名称；pdefaults——返回默认的学习参数；Needg——如果函数使用了 gW 或 gA，则返回 1。

【例 7-38】 learnkt 函数示例。

```
>> p = rand(2,1);
a = rand(3,1);
w = rand(3,2);
lp.lr = 0.5;
dW = learnk(w,p,[],[],a,[],[],[],[],[],lp,[])
```

运行程序，输出结果如下：

```
dW =
   -0.0291    -0.0400
    0.3236    -0.3227
    0.1369    -0.1503
```

（2）learnson 函数

功能：该函数为自组织映射权值学习函数。

其调用格式如下：

```
[dW,LS] = learnsom(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
info = learnsom(code)
```

其参数含义参见 learnk。

在利用该函数进行学习之前，需要设置以下学习参数，见表 7-4。

表 7-4　learnsom 函数默认设置的参数

函数名称	默认值	属性
LP.order_lr	0.9	分类阶段学习速率
LP.order_steps	1000	学习阶段步长
LP.tune_lr	0.02	调谐阶段领域距离
LP.tune_nd	1	调谐阶段学习速率

【例 7-39】learnsom 函数示例。



```
>> p = rand(2,1);
a = rand(6,1);
w = rand(6,2);
pos = hextop(2,3);
d = linkdist(pos);
lp.order_lr = 0.9;
lp.order_steps = 1000;
lp.tune_lr = 0.02;
lp.tune_nd = 1;
ls = [];
[dW,ls] = learnsom(w,p,[],[],a,[],[],[],[],d,lp,ls)
```

运行程序，输出结果如下：

```
dW =
   -0.7607    0.4483
   -0.3594    1.7158
    0.4556    0.3121
    0.1667    0.8575
   -0.3874    0.9287
   -0.2477    0.2897
ls =
        step: 1
      nd_max: 2
```

（3）learnis 函数

功能：该函数为 instar 权值学习函数。

其调用格式如下：

```
[dW,LS] = learnis(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
info = learnis(code)
```

其参数含义参见 learnk。

使用该函数前，需要设置学习速率，如果 LP.lr=0.01，这就是 MATLAB 的默认值，如需调整，只要作相应改动即可。

（4）learnos 函数

功能：该函数为 outstar 权值学习函数。

其调用格式如下：

```
[dW,LS] = learnos(W,P,Z,N,A,T,E,gW,gA,D,LP,LS)
info = learnos(code)
```

其参数含义和学习参数的设定参见 learnis。

5. 初始化函数（midpoint）

功能：该函数为中心点权值初始化函数。

其调用格式如下：

```
W=midpoint(S, PR)
```

其参数说明如下：

- S：神经元的数目。
- PR：每组输入向量的最大值和最小值组成的 $R\times 2$ 维矩阵，规定了输入区间为[Pmin, Pmax]。
- W：函数返回值，$S\times R$ 维的矩阵，每个元素对应设定为（Pmin+Pmax）/2。

6. 权值函数（negdist）

功能：该函数为负距离权值函数。

其调用格式如下：

```
Z = negdist(W,P)
df = negdist('deriv')
```

其参数说明如下：

- W：$S\times R$ 维的以值矩阵。
- P：Q 组输入向量的 $R\times Q$ 维矩阵。
- df = negdist('deriv')：返回值为空，因为该函数不存在导函数。

7. 显示函数（plotsom）

功能：该函数用于绘制自组织特征映射。

其调用格式如下：

```
plotsom(pos)
plotsom(W,D,ND)
```

其参数说明如下：

- pos：S 个 N 维神经元的位置向量。
- W：权值矩阵。
- D：距离矩阵。
- ND：领域矩阵，默认为 1。
- plotsom(pos)：利用红点绘制神经元的位置，将欧氏距离小于等于 1 的神经元连接起来。
- plotsom(W,D,ND)：将欧氏距离小于等于 1 的神经元的权值向量连接起来。

8. 结构函数（hextop）

功能：该函数为六角层结构函数。

其调用格式如下：

```
pos = hextop(dim1,dim2,…,dimN)
```

其参数说明如下：

- dimi：维数为 i 的层长度。
- pos：由 N 个并列向量组成的 $N\times S$ 维矩阵，其中，S= dim1×dim2×…×dimN。

可以利用该函数创建一个二维的神经网络层，共有 40 个神经元，分布在一个 8×5 的六

角格上。

【例 7-40】 hextop 函数示例。

```
>> pos = hextop(8,5); plotsom(pos)
W = rands(40,2); plotsom(W,dist(pos))
```

运行程序，效果如图 7-39 所示。

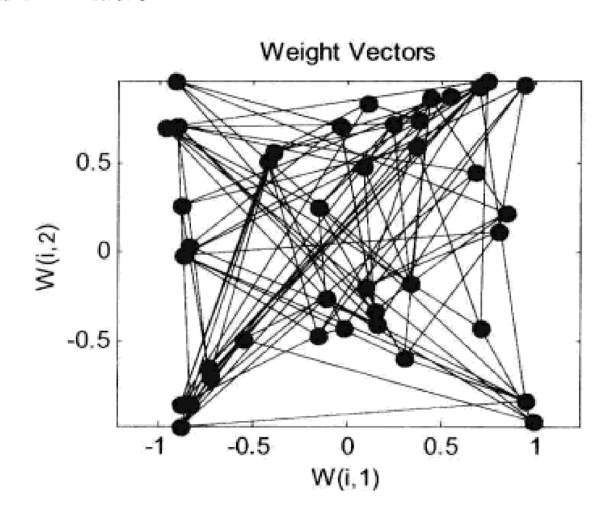


图 7-39 hextop 函数绘图效果



7.7 Simulink 神经网络仿真示例

神经网络工具箱中提供了一套可在 Simulink 中用来建立神经网络的模块。对于在 MATLAB 工作空间中建立的网络，也能够使用 gensim 函数生成一个相应的 Simulink 网络模块。

7.7.1 设置神经网络模块

在 Simulink 库浏览窗口的 Neural Network Blockset 结点上，通过单击鼠标右键后，在弹出的选项中选择“Open Neural Network Blockset Library”命令，便可打开如图 7-40 所示的“Neural Network Blockset”（独立模块集）窗口。

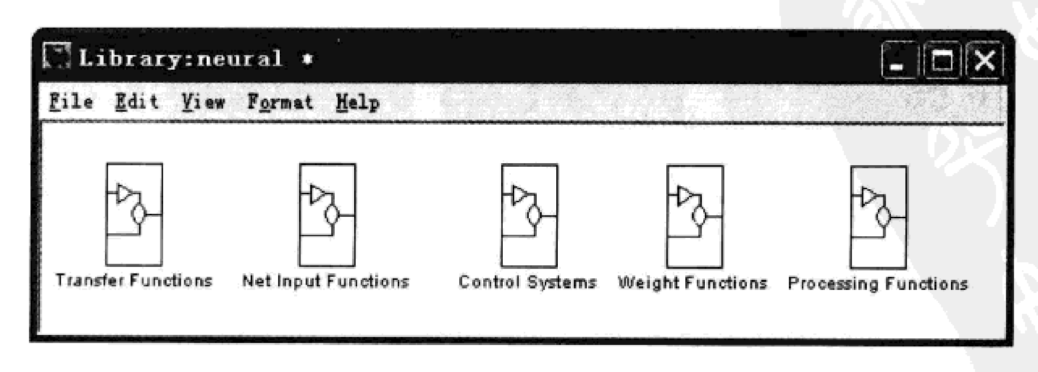


图 7-40 Neural Network Blockset 模块库

在 Neural Network Blockset 模块集中包含了 4 个模块库，用鼠标双击各个模块库的图

标，便可打开相应的模块库。

（1）Transfer Functions（传递函数）模块库

用鼠标双击 Transfer Functions 模块库的图标，便打开如图 7-41 所示的传输函数模块库窗口。传输函数模块库中的任意一个模块都能够接受一个网络输入向量，并且相应地产生一个输入向量，这个输出向量的数组与输入向量相同。

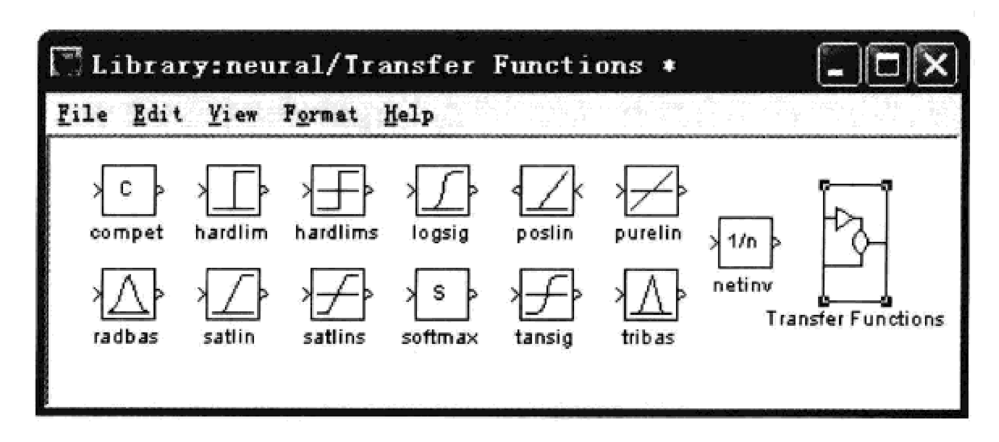


图 7-41　Transfer Functions 模块库窗口

（2）Net Input Functions（网络输入）模块库

用鼠标双击 Net Input Functions 模块库的图标，便可打开如图 7-42 所示的网络输入模块库窗口。

网络输入模块库中的每一个模块都能够接受任意数目的加权输入向量、加权的层输出向量及偏值向量，并且返回一个网络输入向量。

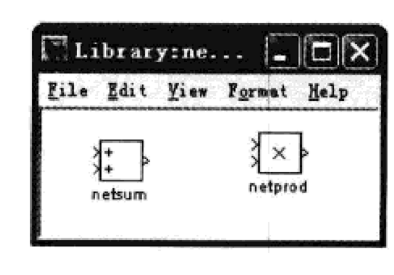


图 7-42　Net Input Functions 模块库窗口

（3）Control Systems（控制系统）模块库

用鼠标双击 Control Systems 模块库的图标，便可打开如图 7-43 所示控制系统模块库窗口。

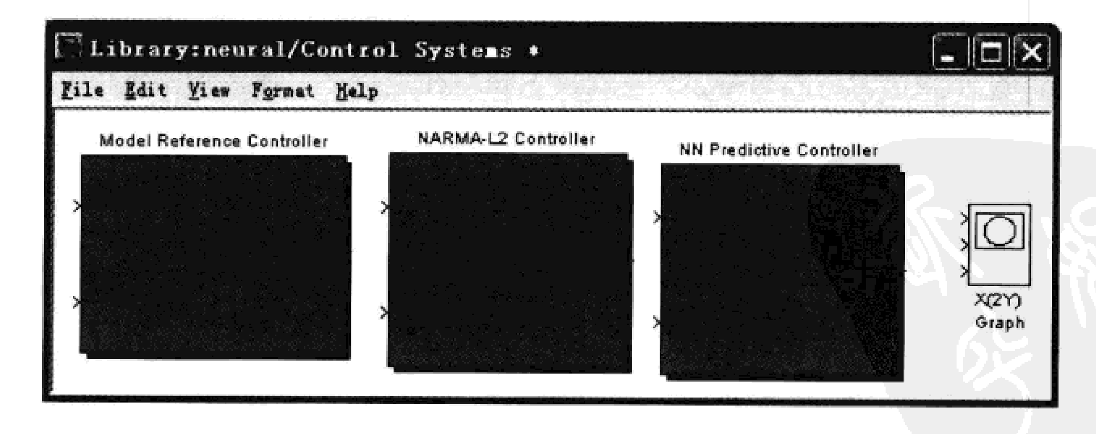


图 7-43　Control Systems 模块库窗口

（4）Weight Functions（权值）模块库

用鼠标双击 Weight Functions 模块库的图标，便可打开如图 7-44 所示的权值模块库窗口。权值模块库中的每个模块都以一个神经元权值向量作为输入，并将其与一个输入向量（或者某一层的输出向量）进行运算，得到神经元的加权输入值。

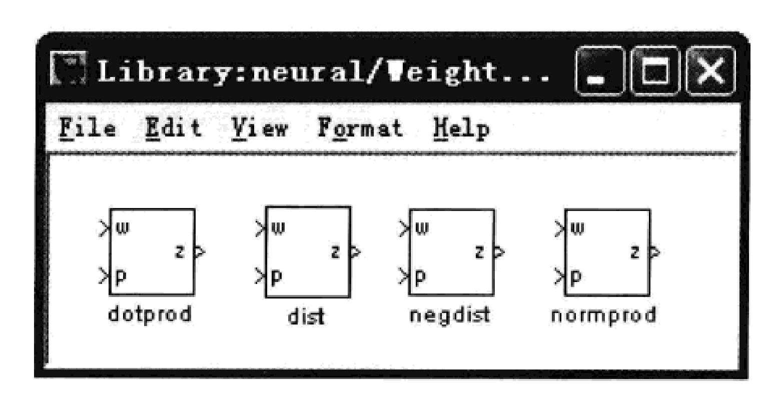




图 7-44 Weight Functions 模块库窗口

（5）Processing Functions（处理函数）模块库

用鼠标双击 Processing Functions 模块库的图标，便可打开如图 7-45 所示控制系统模块库窗口。

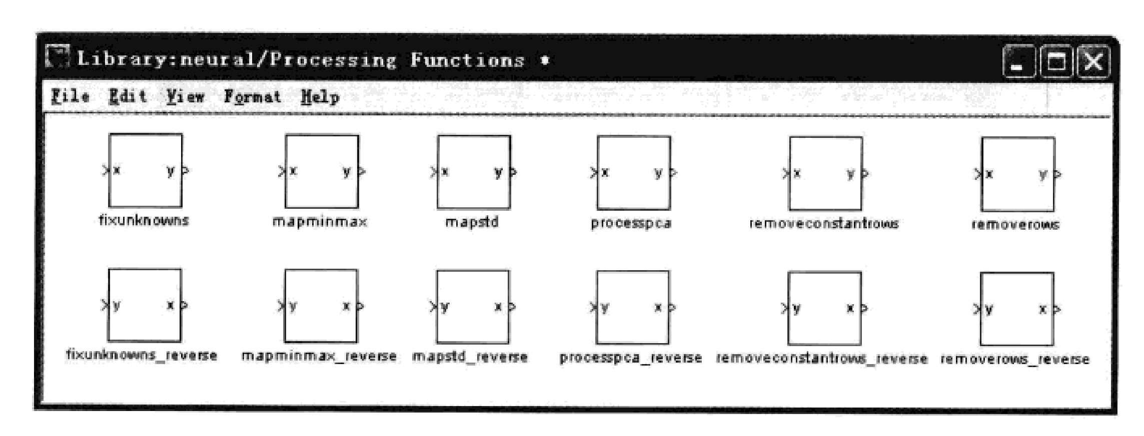


图 7-45 Processing Functions 模块库

7.7.2 神经网络生成模块

在 MATLAB 工作空间中，利用 gensim 函数能够对一个神经网络生成其模块化描述，从而可在 Simulink 中对其进行仿真。

gensim 函数调用格式如下：

```
gensim(net,st)
```

式中，net 参数指定了 MATLAB 工作空间中需要生成模块化描述的网络；st 参数指定了采样时间，它通常情况下为一正数。如果网络没有与输入权值或者层中权值相关的延迟，则指定 st 参数为-1，那么 gensim 函数将生成一个连续采样的网络。

【例 7-41】 设计一个线性网络，并生成其模块化描述。定义网络的输入 X=[1 4 7 2 5]，相应的目标 T= [8 3 6 9 0]。

根据神经网络函数其实现的 MATLAB 程序代码如下：

```
>> X=[1 2 3 4 5];    %给定网络的输入目标值
T= [1 3 5 7 9];   %给定网络的目标值
net=newlind(X,T); %设计一个线性网络
y=sim(net,X)      %利用原始的输入数据测试网络
```

运行程序，输出结果如下：

```
y =
      1.0000     3.0000     5.0000     7.0000     9.0000
```

可以看出，网络已经正确解决了问题。此时可利用以下 MATLAB 命令生成网络的模块化描述。

```
>> gensim(net,-1)    %生成网络 net 的模块化描述
```

以下命令调用函数 gensim 后，将生成一个 Simulink 用户模型窗口，如图 7-46 所示。在这个窗口中包含有一个线性网络，这个线性网络连接了一个采样输入和一个示波器。

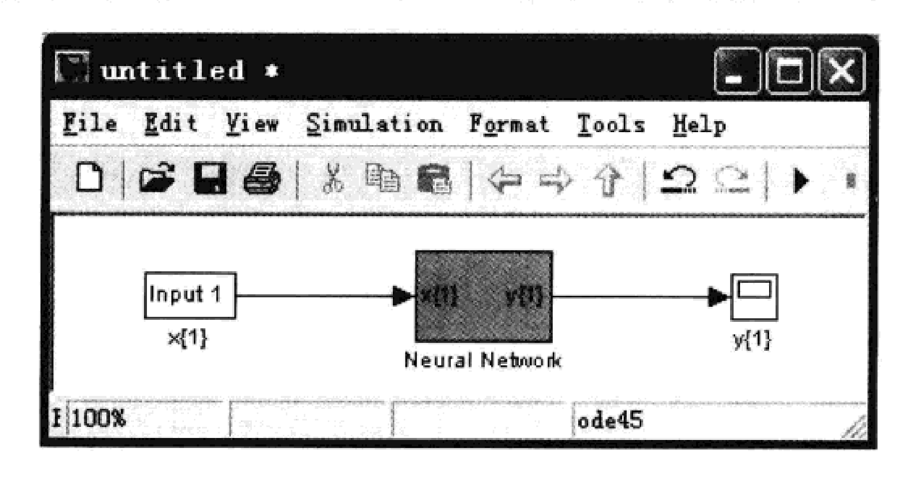


图 7-46 线性网络的 Simulink 仿真图

图 7-46 中的线性网络 net 使用一个神经网络模块（Neural Network）来代替。双击此网络模块，将弹出一个新的窗口，其中绘出了此网络模块的结构，如图 7-47 所示。如果这个结构图还不够具体，不能满足要求，则还可以进一步在其基础上双击需要了解的部分。此时，将继续弹出新的窗口，在此新窗口中出现了网络单击部分的更具体的结构，如图 7-48 所示，为 Layer1 模块的详细结构。这样一直下去，直到出现的窗口为属性设置窗口为止。

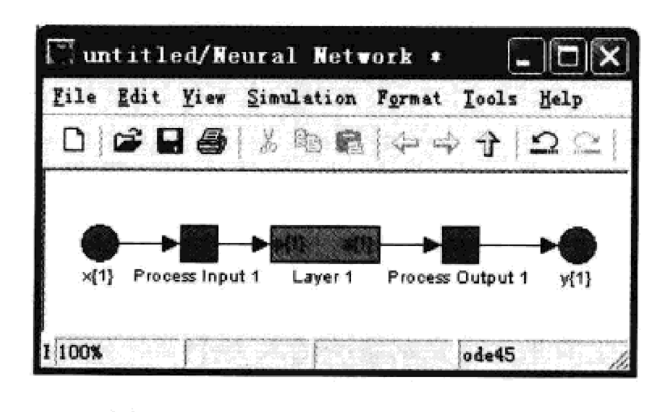


图 7-47 Neural Network 模块的结构

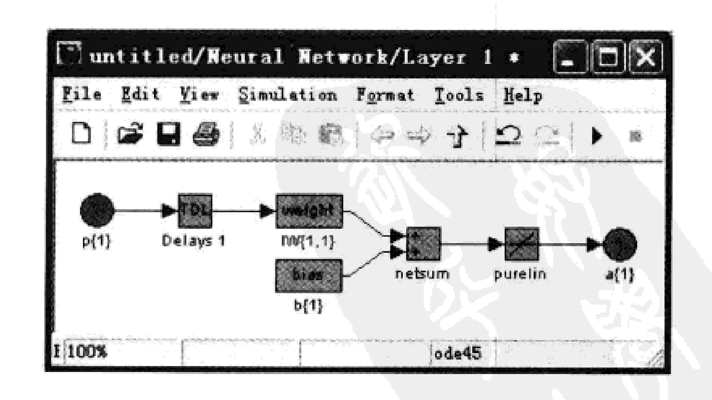


图 7-48 Layer1 模块的详细结构

在图 7-46 所示的窗口中单击 x{1}模块，可打开其属性设置窗口，如图 7-49 所示。该模块实际上是一个标准的常量模块。在窗口中将 Constant value 区域的值改为 5，然后单击“OK”按钮。返回到图 7-46 所示的窗口中，启动仿真，便可在示波器中观察到网络的输出信号，如图 7-50 所示。

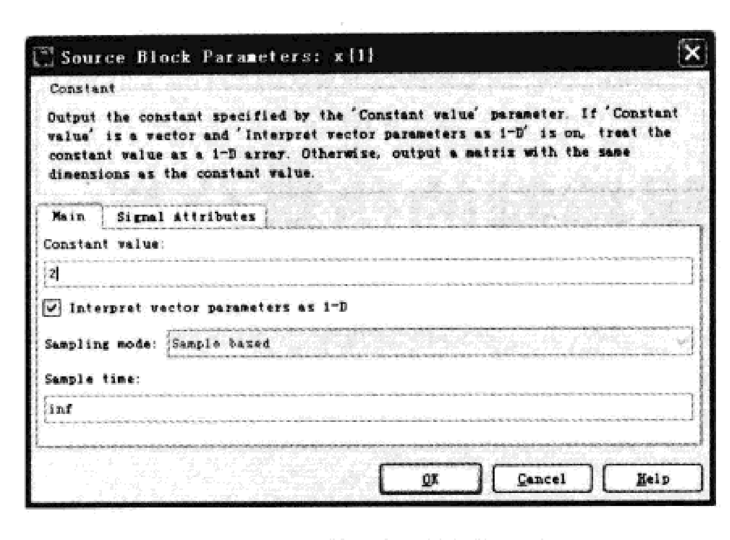


图 7-49　x{1}模块的属性设置窗口

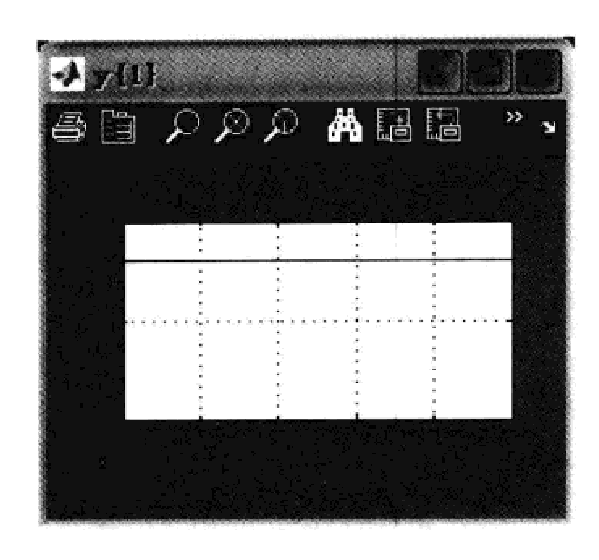


图 7-50　网络的输出信号

从图 7-50 中可看出，网络的输出信号为 3，这与在 MATLAB 工作空间中使用 sim 函数得到的结果是一致的，即输入为 2 时，输出为 3。



第 8 章 模糊逻辑控制的仿真分析

模糊控制是基于丰富操作经验总结出的、用自然语言表述控制策略的，或通过大量实际操作数据归纳总结出的控制规则，用计算机予以实现的自动控制。它与传统控制的最大不同，在于不需要知道控制对象的数学模型，而需要积累对设备进行控制的操作经验或数据。

8.1 模糊逻辑控制概述

1．模糊控制处理的问题

用传统控制方法对一个系统进行控制时，首先要建立控制系统的数学模型，即描述系统内部物理量（或变量）之间关系的数学表达式，必须知道系统模型的结构、阶次、参数等。通常建立系统数学模型的方法有分析法和实验法两种：分析法是对系统各部分的运动机理进行分析，根据它们活动的物理或化学规律列出运动方程；实验法是人为地向系统施加某种测试信号，记录其输出响应，用适当的数学模型去逼近输入/输出间的关系。传统的控制理论都是以被控对象和控制系统的数学模型为基础，进行数学分析和研究的理论。

然而在工程实践中人们发现，有些复杂的控制系统，虽然不能建立数学模型，无法用传统控制方法进行控制，可是凭借丰富的实际操作经验，技术人员却能够通过“艺术性”的操作获得满意的控制效果。

控制对象有很多，它们都相当于一个“黑箱”，由于不知道其中的结构、机理，无法用数学语言描述其运动规律．也就无法建立数学模型，自然无法用传统方法对它实现自动控制。例如，各类窑炉的燃烧过程、有机物的发酵过程、具有非线性强耦合大滞后的复杂系统等，都因为诸如控制对象过于庞大复杂、机理欠明、检测不全、存在大滞后等各种原因而无法建立起清晰的数学模型。然而，人们却可以根据多年的工作经验，把控制它们的操作经验总结成类似上述语言操作规则，按照这些带有模糊性的、用自然语言表述的规则，实现对它们的有效控制。模糊控制基本上解决了用计算模仿人类对这类系统进行的自动控制问题。

2．模糊控制的历程

模糊控制把人类自然语言表述的控制策略，通过模糊集合和模糊逻辑推理转化成数字或数学函数，再用计算机去实现预定的控制。由于模糊控制是以人的操作经验为基础，而不是依赖于控制系统的数学模型，实际上是把人的智能融入了控制系统，自然实现了人的某些智能，所以它属于一种智能控制。

模糊控制的发展历程大致可分为以下 3 个阶段。

（1）形成期（1974 年以前）

1965 年，美国加州大学自动控制系 L.A.Zedeh 教授把经典集合与 J.Lukasiewicz 的多值逻辑融为一体，用数字或函数表达和运算含有像“冷”、“热”之类纯属主观意义的模糊概念，创立了模糊集合理论，这就开创了模糊控制数学基础的研究。其后，出现了许多研究模

糊集合理论和模糊逻辑推理的成果：1968 年提出了模糊算法概念，1970 年提出模糊决策，1971 年提出模糊排序。1973 年，L.A.Zedeh 引入语言变量这一概念，提出用模糊 If-then 规则来量化人类模糊语言的知识规则，建议把模糊逻辑应用于控制领域，从而奠定了模糊控制理论基础。



（2）发展期（1974—1979 年）

1974 年，伦敦大学教授 E.H.Mamdani 博士利用模糊逻辑开发了世界上第一台模糊控制的蒸气机，从而开创了模糊控制的历史。1975 年，英国的 P.J.King 把模糊集合理论应用于反应炉的温度自动控制系统；1976 年，荷兰的 D.V.Nautal Lemke 等人把模糊理论用于多变量非线性控制热水厂热交换过程；1977 年，Mamdani 和 Pappis 把模糊理论应用于马路十字路口的交通管理。

（3）高性能模糊控制阶段（1979 年到现在）

1979 年起，L.P.Holnblad 和 Ostergard 先后在瑞典石灰重烧窑、丹麦水泥窑等工业设备上应用了模糊控制。1983 年，日本富士电机开创了模糊控制在日本的第一项应用——水净化处理。1987 年，日本富士电机致力于模糊逻辑元器件的开发与研究，并在仙台地铁上采用了模糊控制技术；1989 年，日本将模糊控制应用于电冰箱、洗衣机、微波炉等消费产品上，把模糊控制的应用推向了高潮。

很快模糊控制得到了广泛的应用。例如，在炼钢、化工、家用电器、人文社科、经济系统，以及医学心理等领域，要得到正确而且精密的数学模型是相当困难的，但是却具有大量用语言表述的控制规则和只能用语言表述的性能指标，操作人员似乎成了整个系统的组成部分。对于这类问题，靠传统控制方法实现自动控制是非常困难的，但是用模糊控制的方法却可以很容易地进行处理和解决，模糊控制的广泛应用，促进了这项技术的发展。

随着计算机及其相关技术的发展和模糊控制的广泛应用，出现了许多模糊硬件系统，进一步推动了模糊控制理论的发展和应用。模糊控制也由最初的经典模糊控制发展到自适应模糊控制、专家模糊控制和基于神经网络的自学习模糊控制。1992 年，在美国召开了第一届 IEEE 模糊系统国际会议（IEEE International Conference on Fuzzy System）。1993 年，美国 IEEE 神经网络协会创办了国际性模糊专业杂志《Fuzzy System（模糊系统）》，从此模糊控制被人们公认为是智能控制的一个重要分支。

3．模糊控制的特点及展望

模糊控制理论，特别是应用方面在 20 世纪 80 年代末 90 年代初取得了突飞猛进的发展，能被人们广泛接受，是因为它有以下特点。

（1）模糊控制器的设计不依赖于被控对象的精确数学模型

模糊控制是以人对被控对象的操作经验为依据而设计控制器的，故无须知道被控对象的内部结构及其数学模型，这对于传统控制无法实现自动化的复杂系统进行自动控制非常有利。

（2）模糊控制易于被操作人员接受

作为模糊控制核心的控制规则是用自然语言表述的。例如，像"锅炉温度太高，则减少加煤量"这样的控制规则，很容易被操作人员接受，便于进行人机对话。

（3）便于用计算机软件实现

模糊控制规则通过模糊集合论和模糊推理理论，可以转换成数学函数，这样很容易和其他物理规律结合起来，通过计算机软件实现控制策略。

（4）鲁棒性和适应性好

通过专家经验设计的模糊规则，可以对复杂被控对象进行有效的控制，经过实际调试后其鲁棒性和适应性都容易达到要求。

模糊控制是一种反映人类智慧的智能控制方法，由于它的适应面广和易于普及，使它成为智能控制领域最活跃、最重要和最实用的分支之一。尤其是它作为传统控制的补充和改进方法，常与传统控制相结合被应用各种复杂系统的自动化中。目前已经在工业控制及其他领域，诸如炼钢、化工、人文系统、经济系统及医学心理系统中，特别是家用电器自动化领域和其他很多行业中解决了传统控制方法无法或者难以解决的实际问题，取得了令人瞩目的成效，引起越来越多的控制理论研究人员和相关领域的广大工程技术人员的极大兴趣。

但是，我们也应该看到，就目前的状况来看，模糊控制尚缺乏重大的理论性突破，无论在理论上，还是在应用上都有待进一步的深入研究和探讨。特别是在下述几个方面：

1）需要对模糊系统的建模、模糊规则的确立和模糊推理方法等进行深入研究，特别是对于非线性复杂系统的模糊控制。

2）模糊控制系统的创建和分析方法仍停留在初级阶段，稳定性理论还不成熟，这些都需要进一步探讨。

3）需要进一步开发和推广简单、实用的模糊集成芯片和通用模糊系统硬件。

4）需要对模糊控制系统的设计方法加强研究，把现代控制理论、神经网络与模糊控制进行更好的结合、相互渗透，在多方面进行深入研究，以便构成更多、更好的模糊集成控制系统。

模糊控制理论的提出，是控制思想领域的一次深刻变革，它标志着人工智能发展到了一个新阶段。特别是对那些时变的、非线性的复杂系统，在无法获得被控对象清晰数学模型的时候，利用具有智能性的模糊控制器，可以给出较为有效的自动控制方法。因此，模糊控制既有广泛的实用价值，又有很大的发展潜力。

8.2 模糊逻辑工具箱的图形界面

模糊控制系统和经典控制系统的总体结构是类似的，尤其是和 PID 控制系统之间，只是控制器不同而已。模糊控制器模糊控制系统的核心，是一个典型的模糊推理系统。

MATLAB 的 Simulink Library Browser（仿真模块库）中，设有专用的模糊逻辑工具箱（Funzzy Logic Toolbox），它提供了用于模糊逻辑系统的命令行（在 Command Window 中使用）和图形用户界面（Graphical User Interfaces，GUI）两种仿真方式。两者都可以方便地建立、编辑、观察、分析和设计模糊推理系统（Fuzzy Interence System，FIS），进行模糊推理系统的仿真。模糊控制器可以说是一类用途特殊的模糊推理系统，它是在模糊系统中用作控制器的模糊推理系统。如 T-S 型模糊推理系统不仅可以用作模糊控制器，而且可以逼近任意非线性系统，适用于任意模糊系统。

8.2.1 模糊推理系统图形用户界面介绍

在 MATLAB 中，模糊推理系统的 GUI 是进行模糊系统仿真的重要工具，尤其是设计、建立、仿真和分析模糊控制器，用它显得特别简捷、直观和经济。模糊推理系统的 GUI，由（Rule Editor、FIS Editor、Member ship Function Edito、Surface Viewer、Rule Viewer）5 个界

面组成。

在 FIS 的 GUI 5 个界面中，3 个是可以互动的编辑器：

1）Rule Editor（模糊规则编辑器）。

2）FIS Editor（模糊推理系统编辑器）。

3）Member ship Function Edito（隶属函数编辑器）。



用户在这 3 个编辑器中，可以完成 Mamdani 型和 Sugeno 型两类模糊推理系统的结构编辑、模糊子集的隶属函数及其分布的选定、模糊规则的建立等主要的设计任务，以及控制效果的仿真观测和设计参数的调试。这 3 个编辑器是可以互动的，用户随时可以打开任何一个编辑器进行删改、编修、调试、观测和保存。同时，这 3 个编辑器又是联动的，只要在一个编辑器中进行编修，另两个就会自动作出相应的变更。

另外两个界面属于只读工具，提供的是只供查看用的观测窗：

1）Surface Viewer（输出量曲面观测窗）。

2）Rule Viewer（模糊规则观测窗）。

在前面 3 个编辑器内完成的编辑工作，其效果可以在后两个观测窗中查看，根据显示情况进行分析、研究，提出修缮意见，重返编辑器进行改进。虽然每个界面的功能不尽相同而且是相对独立的，却又相互关联而动态连接着，在任何一个窗口界面上更改参数或性态，打开其他几个窗口界面时，相关参数和性态都会自动地做相应的变更。

8.2.2 模糊推理系统编辑器介绍

在 3 个编辑器中，模糊推理系统编辑器（FIS Editor）是关乎模糊系统框架、主体结构等总体大局设计的编辑器，它可以编辑、设计、修改整个系统结构，增减系统输入、输出变量的个数，调整模糊系统的维数。我们介绍的 Mamdani 型和 Sugeno 型模糊推理系统，其基本结构相同的，都可以在这里完成。因此，设计任何模糊系统，都应该先用 FIS 编辑器设计完成系统的总体结构之后，再分别进行细目编辑与设计，最终再返回修改、调整和完善。

启动 MATLAB 后，在主窗口输入 fuzzy 并按回车键，屏幕上就会显示如图 8-1 所示的“FIS Editor”界面，即模糊推理系统编辑器。

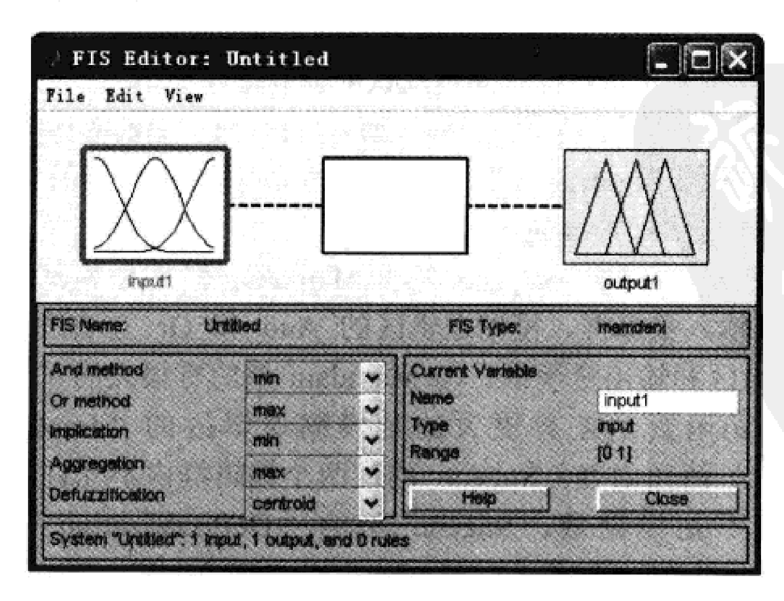


图 8-1 FIS (Mamdani)编辑器设计界面

1．FIS 编辑器界面简介

FIS 编辑器是设计模糊推理系统的重要辅助工具，它的界面分为上、下两部分。

1）界面上部设有菜单条和模框区，利用它们可以编辑模糊推理系统的输入、输出变量和模糊控制规则。

菜单条有 3 个主菜单，分别是“File”（文件）、“Edit”（编辑）和“View”（视图）。

单击“FIS”编辑器上任何一个主菜单，可以打开该主菜单下属的子菜单。利用这些菜单，配合模框可以设计 FIS 的模糊推理类型、模糊系统的结构、维数、变量数目等内容。

模框区设有 3 个模框，左边是“输入变量模框”，右边是“输出变量模框”，中间是“模糊规则编辑模框”。双击其中之一，就可以打开相应的编辑器，设计输入变量、输出变量和进行模糊规则编辑。

2）界面下部主要包含模糊逻辑区和当前变量区。

左边的“模糊逻辑区”，显示模糊逻辑运算、模糊推理、综合和清晰化等方法。利用下设的条目可以编辑模糊逻辑运算方法、综合各控制规则结论的方法和模糊结论清晰化方法。

右边的“当前变量区”显示变量名称、类型、显示范围及其相应的编辑框。利用该区的项目可以编辑输入/输出变量的名称、类型、显示范围。

当前变量区下边，左侧设有“Help”（提供帮助）和“Close”（关闭当前界面）两个按钮。单击“Help”按钮，可以查寻界面内任何内容的帮助说明，单击“Close”按钮则关闭窗口。

上、下部之间有一行是“系统状态显示行”，图 8-1 界面上显示：“FIS Name untitled FIS Type：mamdani（FIS 名称 未定 FIS 类型 曼达尼）”，表明当前系统的状态。

FIS 编辑器界面最下一行为系统“结构显示行”，图 8-1 界面上显示：“System ‘Untitled’：1 input，1 output，0 rules（系统‘未定’：1 个输入，1 个输出，0 条规则）”，显示出当前系统的结构。

2．FIS 推理类型的编辑

利用 FIS 编辑器界面上的菜单，可以设计有关模糊推理系统的各项结构性内容。

MATLAB 中设有两种类型的模糊逻辑推理：Mamdani（曼达尼）和 Sugeno（苏杰瑙，即 T-S）。当前推理类型显示在“模糊规则编辑模框”中，在图 8-1 中为“(mamdani)”，表明当前属“曼达尼”型模糊推理，这是系统默认的模糊推理类型。

要想把模糊推理类型改为 T-S（Sugeno）型，可在 FIS 编辑器界面单击“File”菜单下的“New FIS…”选项下的“Sugeno”命令，系统弹出如图 8-2 所示的 T-S 型 FIS Editor，中间模糊规则编辑模框内显示有“(sugeno)”。

比较图 8-1 和图 8-2 可知，Sugeno 型和 Mamdani 型 FIS 编辑器的界面结构大体相同，它们的主菜单、输入量模框、模糊逻辑区的 And 和 Or 方法是完全一样的。但是，Sugeno 型的模糊逻辑区和输出量模框，与 Mamdani 型的有很大的差异，输出量显示的不是模糊子集而是 $f(u)$ 函数。差异更大的是模糊逻辑区的“Implication”（蕴涵）和“Aggregation”（综合）两项内容，Sugeno 的这两个编辑框内不允许填入内容，因为它输出的结论是函数而不是模糊量。Sugeno 型的“Defuzzification”（清晰化）相当于 Mamdani 推理中“Implication”、“Aggregation”及“Defuzzification”三者的综合结果，所以编辑框内的预设选项跟 Mamdani 的也大不相同，仅为“Wtaver”（加权平均）和

"Wtsum"（加权求和）。

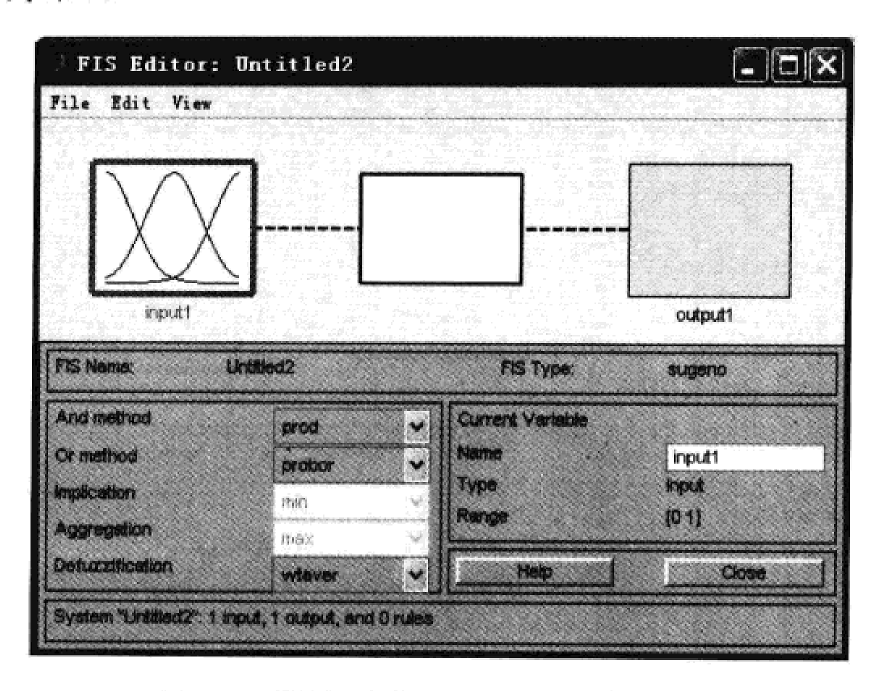


图 8-2　模糊推理系统（Sugeno）编辑器界面



3. 编辑 FIS 的维数

图 8-1 和图 8-2 显示的都是单输入单输出（SISO）一维模糊推理系统。工业上应用得最多的是二维模糊控制器，即同时把一个变量和它的变化率输入控制器，去调节输出量。下面介绍增、减控制器维数的方法。

1）在 FIS 编辑器界面（Mamdani 型或 Sugeno 型）上，单击"Edit"菜单下的"Add Variable..."选项下的"Input"命令，就变成 SISO 二维模糊推理系统。

当这个操作在图 8-1 的 FIS 编辑器（Mamdani）界面上进行时，系统弹出如图 8-3 所示的 SISO 二维 Mamdani 型模糊系统。

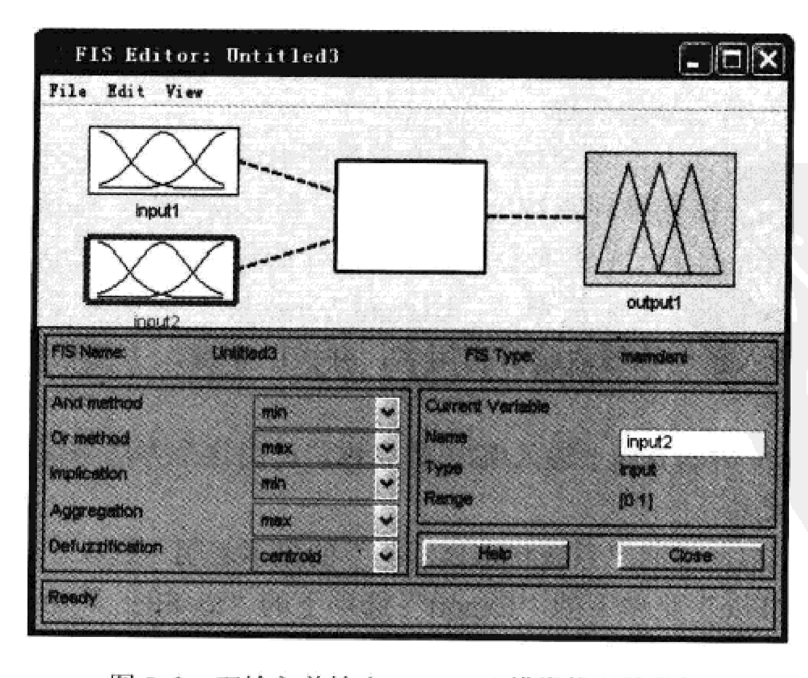


图 8-3　双输入单输出 Mamdani 模糊推理编辑器

当这个操作在图 8-2 的 FIS 编辑器（Sugeno）界面上进行时，系统弹出如图 8-4 所示的 SISO 二维 Sugemo 型模糊系统。

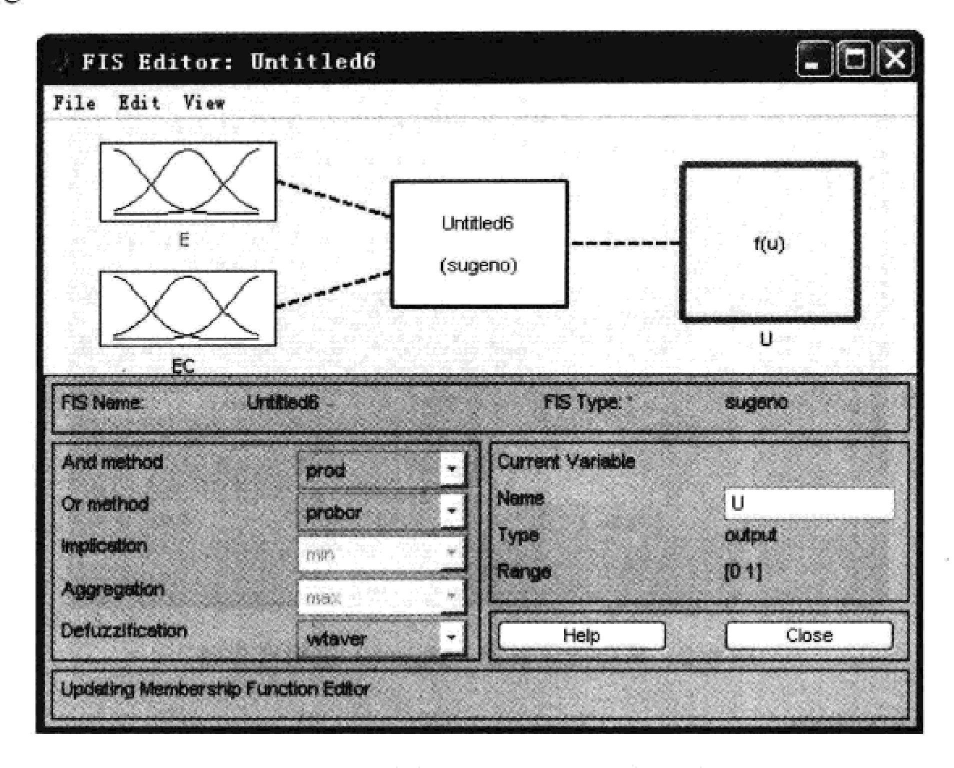


图 8-4 双输入单输出 Sugeno 模糊推理编辑器

比较图 8-3 和图 8-4 界面可知，它们的差异依然在输出变量模框和模糊逻辑区上。

2）要想增添一个输出变量，单击“Edit”菜单下的“Add Variable...”选项下的“Output”命令。

3）如果想删去一个已有的输入或输出变量，可先选中待删减变量的模框，再单击“Edit”菜单下的“Remove Selected Variable”（移去选定的变量）命令，即可。

多次重复上述增、减输入/输出变量的操作，可使控制器的维数不断变化，直到满足设计要求。

4. 编辑 FIS 输入/输出量的名称

新打开的 FIS 编辑器上，各个输入/输出变量的名称都是临时的暂用名，如 input1、output2 等。在针对具体模糊推理系统的设计时，需要更改它们。具体步骤如下：

1）在 FIS 编辑器界面上，单击需要重新命名变量的模框，使它的边框线变组、变红。

2）在 FIS 编辑器界面的当前变量区“Current Variable”（当前变量），单击“Name”右侧编辑框，从键盘输入新名称覆盖临时暂用名，并按回车键，上部相应模框下的名称也跟着被更改。

图 8-4 中的 FIS 编辑器界面上输入/输出量名称，已用这个方法分别改成 E、EC 和 U。

5. 编辑 FIS 的名称

图 8-1 和图 8-2 界面中间“系统状态显示行”的显示表明，这 4 个 FIS 都未被命名。命名的方法是：单击“File”菜单下的“Export”选项下的“to File...”命令，系统弹出“Save FIS”（保存 FIS）界面，如图 8-5 所示。

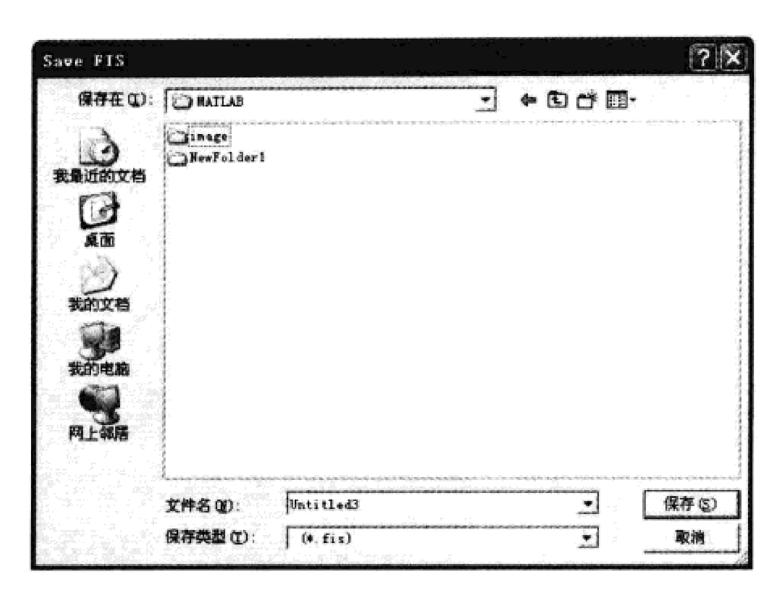



图 8-5　保存 FIS 界面



在该界面下部“文件名（N）”右侧的编辑框内，填入 FIS 的新名称覆盖掉“Untitled”，再单击界面右下角的功能按钮“保存（S）”。文件就以新的名称被保存在“MATLAB”子目录中，当然也可以存入其他子目录或自己的 U 盘中。

6．编辑模糊逻辑推理的具体算法

在选定模糊推理类型 Mamdani 或 Sugeno 之后，还需选择每种推理类型中模糊逻辑的具体算法。这些编辑工作主要在 FIS 编辑器下部“模糊逻辑区”内进行。由于不同类型模糊逻辑推理的具体算法不尽相同，下面分别予以介绍。

（1）Sugeno 型模糊逻辑算法

图 8-4 是一个 Sugeno 型模糊推理系统编辑器。该界面下部的模糊逻辑区中，含有“And Method”（与方法）、“Or Method”（或方法）和“Defuzzification”（清晰化）3 项模糊逻辑运算方面的内容。每一项的右侧都有一个“编辑框”，单击它可弹出几种备选模糊逻辑算法。

（2）Mamdani 型模糊逻辑算法

图 8-3 是一个 Mamdani 型模糊逻辑推理系统编辑器。该界面下部模糊逻辑区中，设有“Add”（与）、“Or”（或）、“Implication”（蕴涵）、“Aggregation”（综合）和“Defuzzification”（清晰化）5 项模糊逻辑运算方面的内容。前 3 项都是构成复合模糊命题的连接词；第 4 项“Aggregation”是多条模糊规则结论被“合并综合”时用的算法；第 5 条是清晰化方法。每项内容的右侧都是有一个“编辑框”，单击任意一项编辑框，就下拉出预先设置的具体逻辑算法。

8.2.3 隶属度函数编辑器介绍

1．MF 编辑器界面介绍

在 FIS 编辑器界面上，双击输入量或输出量模框中的任何一个，都会弹出隶属函数编辑器（Membership Function Editor），简称 MF 编辑器。若在图 8-1 所示界面上进行这一操作，如图 8-6 所示。

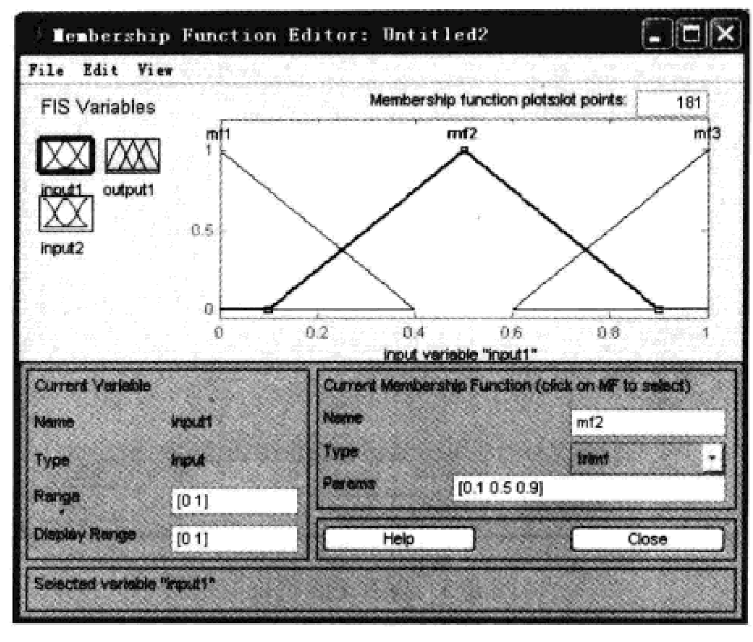


图 8-6 Mamdani 型变量 MF 编辑器

1）MF 编辑器上有 3 个主菜单，其中 File（文件）和 View（视图），与 FIS 编辑器中的功能、用法完全一样。另一个主菜单 Edit（编辑）之下有 7 个子菜单，它们的主要功能都与编辑隶属函数有关，其中子菜单 Undo 的功能是撤销刚才的操作，跟其他编辑器中的功能、用法完全一样。

2）MF 编辑器界面上部左侧为“变量模框索引区”，右侧为“图形函数区”。单击模框索引区中任何一个小模框，都会使它的边框变红、变粗，同时图形函数区内显示出相应的函数图形或函数名称。

3）MF 编辑器界面下部左侧为“Current Variable”（当前变量区），界面下部右侧为“Current MF”（当前隶属函数区）。它们之下预设有许多选项。

在 MF 编辑器中，可以编辑覆盖模糊论域的子集数目、调整模糊子集的分布、选定各模糊子集的名称及隶属函数类型等。

由于 Sugeno 型 FIS 的 MF Editor 编辑器界面的结构和图 8-6 所示的 Mamdani 型的一样，在此不再介绍。

2. Mamdani 型 FIS 中隶属函数（MF）的编辑器

（1）编辑输入变量的论域和显示范围

在图 8-6 所示的 MF 编辑器（Mamdani）界面上，单击“变量模框索引区”中待编辑变量的小模框，使其边框变红、变粗，则界面下部“当前变量区”内就显示出该变量的性态，以供编辑。

（2）增加覆盖输入量模糊子集的数目

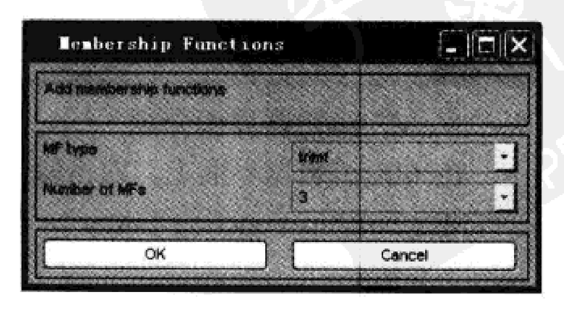


图 8-7 “MF 编辑器”对话框

新打开的 MF 编辑器界面上，“图形函数区”显示有 3 个默认的模糊子集（隶属函数），覆盖着设定的显示范围，根据设计需求可以增添覆盖变量论域的模糊子集个数。增添的方法是在 MF 编辑器上，单击“Edit”菜单下的“Add MFs”命令（或单击“Edit”菜单下的“Add Custom MF...”命令），则弹出如图 8-7

所示的“MF 编辑器”对话框。

图 8-7 界面上第一行显示出对话框的功能“Add membership functions”（添加隶属函数），编辑 MF 的数量和类型。界面中间有两项可编选的内容：“MF type”（MF 类型）和“Number of MFs”（隶属函数的数目），可通过单击其右边的下拉列表框，进行设置。



（3）编修隶属函数曲线

初步设定的覆盖模糊论域隶属函数个数、类型，在编修中经常需进一步的细化修缮。如对隶属函数名称的重新命名、函数类型的异化和筛选、位置的排布等编修工作。

（4）编修模糊子集位置

模糊子集在变量论域上的分布，即散布方式，必须按完备性、一致性和交互性进行排列。用鼠标可以移动模糊子集在显示范围中的相对位置，移动方法也是用拖法或参数法，无论用哪种方法，都是设法改变隶属函数的核及其子集在显示范围中的相对位置。

例如，在图 8-6 所示的 MF 编辑器图形框中，模糊子集 mf1 是三角形隶属函数，拐点的参数为[-0.4 0 0.4]。改变这个模糊子集的相对位置，主要是变动隶属函数的核或取值最大点的位置，这里就要变动 0。当然，有时为了某种特殊需求，也要改变隶属函数支集端点-0.4 和 0.4 的位置。

（5）删除模糊子集的方法

修改设计中，常常需要删除已有的模糊子集，可按下述步骤进行。

1）单击要删除的模糊子集隶属函数曲线，使其变红变粗。

2）在 MF 编辑器界面上，单击“Edit”菜单下的“Remove selected MF...”选项，该曲线则被删掉。相应地，模糊子集也被删掉。也可以在选中隶属函数后，按“Delete”按钮删除。

对于 Mamdani 型 FIS 中输出变量的模糊子集（隶属函数），编修方法跟上述的编修输入量模糊子集的方法完全一致，不再介绍。

3. Sugeno 型 FIS 中隶属函数（MF）的编辑

Sugeno 型模糊推理和 Mamdani 型模糊推理，其输入量的模糊化处理方法完全一样，所以覆盖输入量隶属函数的编辑工作也相同，编辑模糊子集的论域、显示范围、模糊子集数的添减等方法也完全相同，这里不再介绍。

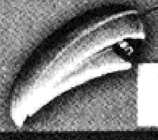
两种类型推理的输出结论大不相同，Mamdani 型模糊推理输出的是模糊子集，而 Sugeno 型模糊推理输出的是线性函数，下面介绍在 Sugeno 型模糊推理中，输出函数的编辑方法。

（1）进入二维 Sugeno 型 FIS 编辑器

在双输入单输出 FIS 编辑大（sugeno）界面上，将输入、输出变量的名称分别改为 E、EC 和 U，如图 8-4 所示。

（2）调出 Sugeno 型 MF 编辑器

在如图 8-4 所示的 Sugeno 型 FIS 编辑器界面上，双击任何一个变量模框，系统弹出 Sugeno 型 MF 编辑器，再在模框索引区中单击输出模框，变得出 Sugeno 型输出量 MF 编辑器界面，如图 8-8 所示。



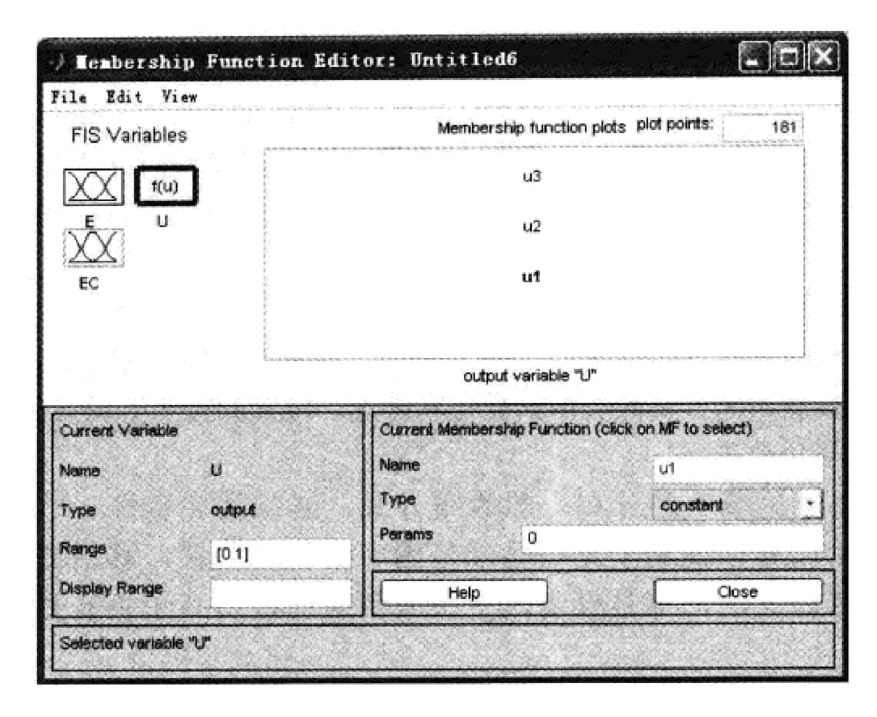


图 8-8 Sugeno 型 MF（输出量）编辑器

8.2.4 模糊规则编辑器

模糊规则就是输入量和输出量间的模糊蕴涵关系 R，只是用 F 条件命题对它们进行了表述。编辑模糊规则之前，必须首先完成模糊系统的结构、模糊推理的类型和输入变量的模糊化编辑。下面介绍用模糊规则编辑器编辑模糊规则的方法。

1. 模糊规则编辑器界面简介

Sugeno 型和 Mamdani 型模糊规则编辑器界面的结构形式是一样的，这里仅以 MATLAB 中的模糊型仿真示例“tank”为例，介绍 Mamdani 型 FIS 的模糊规则编辑器。

在 GUI（TANK）的任何一个编辑器界面上，单击“Edit”菜单下的“Rulles...”命令，系统弹出如图 8-9 所示的模糊规则编辑器（Rule Editor：tank）界面，简称 Rule 编辑器。

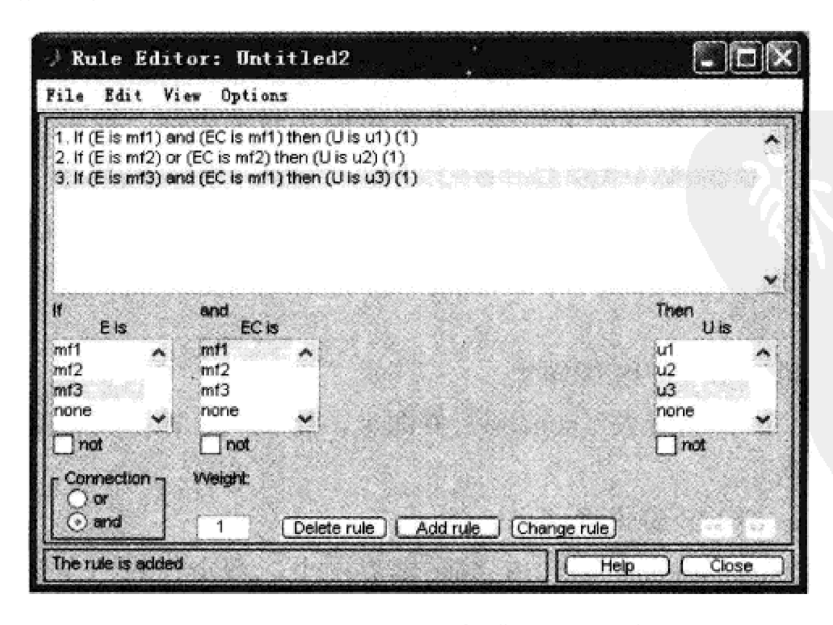


图 8-9 Mamdani 型 FIS 的模糊规则编辑器



2. 表述模糊规则的语言和格式编辑

模糊规则可以说是用“语言”表述的“微分方程”。因此，模糊规则的编辑中离不开“语言”编辑，Rule 编辑器中预先设置 3 种表述模糊规则的语言。通常在 Rule 编辑器界面上，首选的默认语种是“英语”，通常不用德语和法语，不必再进行编辑。

Rule 编辑器预先设置 3 种模糊规则的表述格式，它们各有优劣，可以自动转换。在 Rule 编辑器界面上，单击“Options”菜单下的“Format”选项，系统弹出表述模糊规则的 3 种格式，再单击选用的格式，就成为当前表述模糊规则的格式。

模糊规则编辑器的菜单功能与模糊推理系统编辑器基本类似，在其视图菜单中能够激活其他的编辑器或窗口。

界面下部还有 3 个按钮，分别为删除规则（Delet rule)、增加规则（Add rule）及修改规则（Change rule)。

在这个界面下编辑模糊规则是十分方便的，系统已经自动地将在 FIS Edit 中定义的变量显示在界面的左下部。在窗口上选择相应的输入变量（以及是否加否定词 not)，然后选择不同变量之间的连接关系（or 或 and）以及输入权重（默认为 1)，然后单击“Add rule”按钮，我们刚刚输入的规则已经在编辑器上面的显示区域中出现了。

例如，在 *E* 变量中选择 mf1，在 *EC* 变量中选择 mf1，在“Connection”选项中选择 and，单击“Add rule”按钮，结果出现：

1. If (E is mf1) or (EC is mf1) then (U is u1) (1)

括号中的数字是该规则的权重值。

这里我们加入如下全部 3 条规则（权重均为 1)：

- 1. If (E is mf1) or (EC is mf1) then (U is u1) (1)。
- 2. If (E is mf2) or (EC is mf2) then (U is u2) (1)。
- 3. If (E is mf3) and (EC is mf1) then (U is u3) (1)。

完成之后，在图 8-5 界面上部的白色区域内可以观察到刚加入的模糊推理规则，可以从菜单“Options”项目中选择相应的显示语言和显示方式。

将显示方式设为 symbolic，显示变为：

- 1. (E==mf1) & (EC==mf1) => (U=u1) (1)。
- 2. (E==mf2) | (EC==mf2) => (U=u2) (1)。
- 3. (E==mf3) & (EC==mf1) => (U=u3) (1)。

如设为 indexed，显示将变为：

1 1, 1 (1) : 1

2 2, 2 (1) : 2

3 1, 3 (1) : 1

虽然显示的方式不同，甚至可能没有 if、then 这样的词，但这些规则内部实际的含义仍然是相同的。

3. 删除编好的模糊规则

要删除某条模糊规则，先单击该规则，使其背景变暗；再单击界面下部的“Delete rule”功能按钮，这条模糊控制规则就被删除，从“显示区”中消失。

8.2.5 模糊规则观测窗

模糊逻辑 GUI 的 5 个界面中，规则观测窗和输出曲面观测窗是只读性的，没有编辑功能，也不能互动，只供观测用。规则观测窗用图形界面形象地显示出模糊推理的过程，在编辑完成模糊子集、模糊规则及推理方法等内容之后，可以调出它来进行观测。无论在 GUI 的哪个界面中，都可通过单击“View”菜单下的“Rules”选项，系统弹出如图 8-10 所示的规则观测窗口。

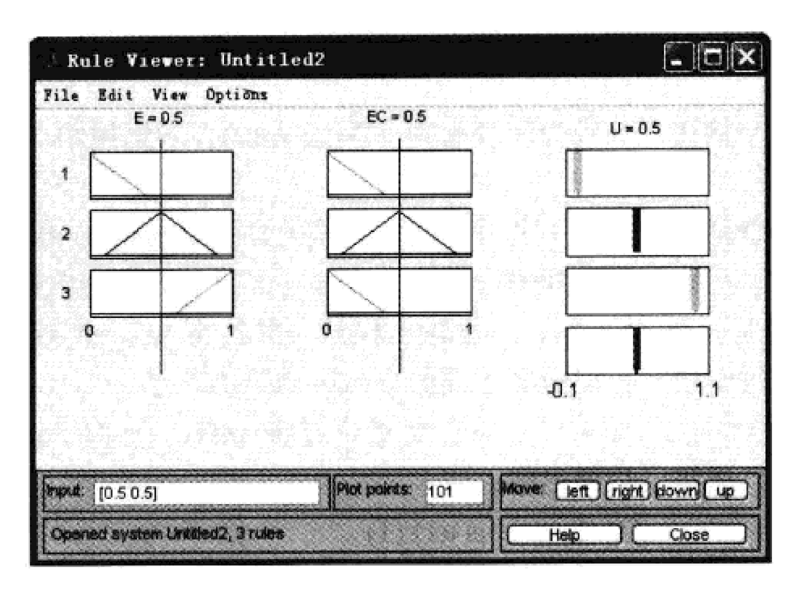


图 8-10 模糊规则观测窗口

8.2.6 模糊推理输入/输出曲面观察器

在模糊推理系统编辑窗口、隶属函数编辑窗口、模糊规则编辑窗口中单击“View”菜单下的“Surfview”选项，系统弹出如图 8-11 所示的输入/输出曲面观察器。

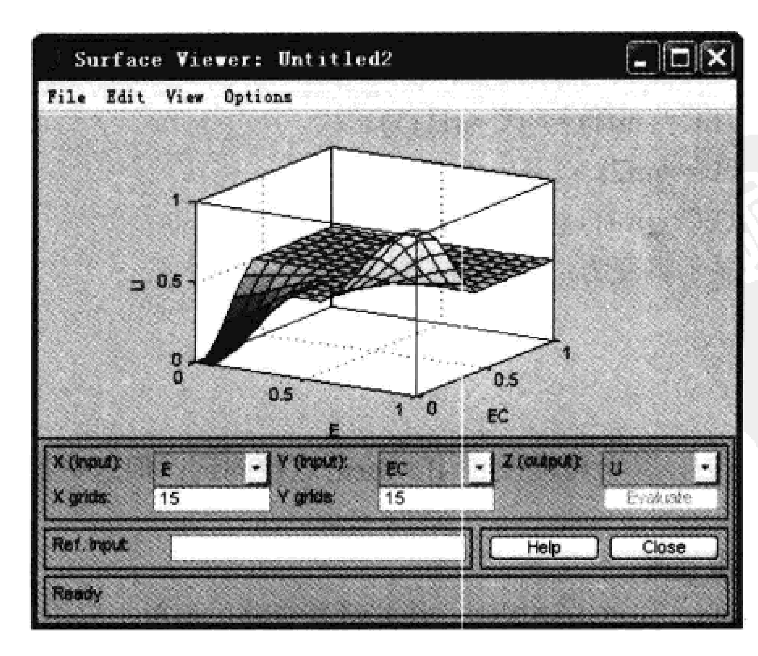


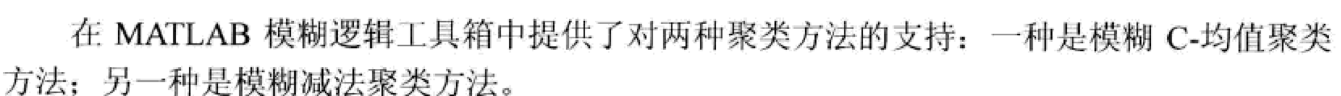
图 8-11 输入/输出关系曲面观察器

8.3 模糊聚类



对给定的数据，聚类分析是许多分类和系统建模问题的基础。聚类的目的是从大量的数据中抽取固有的特征，从而获得系统行为的简洁表示。常见的聚类方法有均值聚类、分层聚类和减法聚类等。

在 MATLAB 模糊逻辑工具箱中提供了对两种聚类方法的支持：一种是模糊 C-均值聚类方法；另一种是模糊减法聚类方法。



8.3.1 模糊 C-均值聚类函数

在模糊 C-均值聚类方法中，每一个数据点按照一定的模糊隶属度隶属某一聚类中心。这一聚类技术作为对传统聚类技术的改进，是 Jim Bezdek 于 1981 年提出的。该方法首先随机选取若干聚类中心，所有数据点都被赋予对聚类中心一定的模糊隶属度，然后通过迭代方法不断修正聚类中心，迭代过程以极小化所有数据点到各个聚类中心的距离与隶属度值的加权和为优化目标。

模糊 C-均值聚类的输出不是一个模糊推理系统，而是聚类中心的列表及每个数据点对各个聚类中心的隶属度值。该输出能够被进一步用来建立模糊推理系统。对应模糊 C-均值聚类方法的函数为 fcm，该函数调用格式如下：

```
[center,U,obj_fcn] = fcm(data,cluster_n)
```

式中，输入参数 data 为给定的数据集；矩阵 data 为每一行为一个数据点向量；cluster_n 为聚类中心的个数；center 为迭代后得到的聚类中心；U 为所有数据点对聚类中心的隶属函数矩阵；obj_fcn 为目标函数值在迭代过程中的变化值。

聚类过程在达到最大次数或满足停止误差准则时结束。

尽管 fcm 函数的输出不是一个模糊推理系统，而是聚类中心的列表及每个数据点对各个聚类中心的隶属度值，但该函数的输出能够进一步用来建立模糊推理系统。

【例 8-1】 利用模糊 C-均值聚类法将一组随机给定的二维数据分为两类。

其实现的 MATLAB 程序代码如下：

```
>>data = rand(100, 2);
[center,U,obj_fcn] = fcm(data, 2);
plot(data(:,1), data(:,2),'o');
maxU = max(U);
index1 = find(U(1,:) == maxU);
index2 = find(U(2, :) == maxU);
line(data(index1,1), data(index1, 2), 'linestyle', 'none',
'marker', '*',
'color', 'g');
line(data(index2,1), data(index2, 2), 'linestyle', 'none',
'marker', '*',
'color', 'r');
title('模糊 C-均值聚类');
```

运行程序，输出效果如图 8-12 所示。

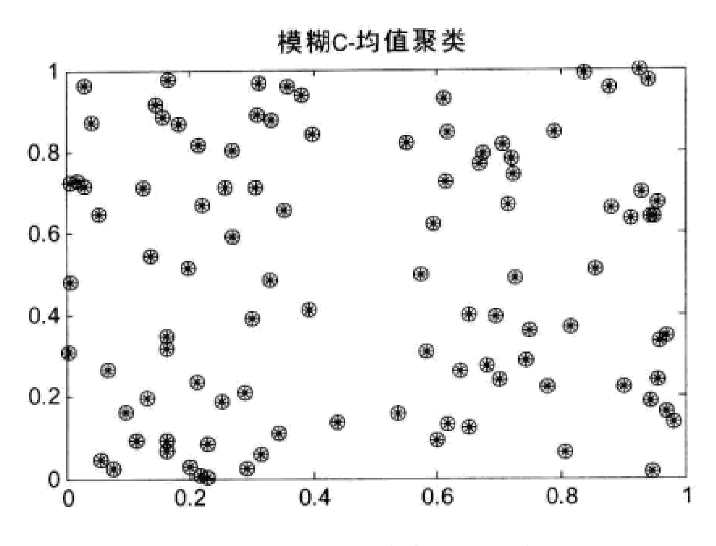


图 8-12　模糊 C-均值分类结果

8.3.2 减法聚类

减法（Subtractive）聚类是一种用来估计一组数据中的聚类个数及聚类中心位置的快速的单次算法。由减法聚类算法得到的聚类估计可以用于初始化那些基于重复优化过程的模糊聚类及模型辨识方法，如自适应神经模糊系统的算法函数 anfis。

模糊减法聚类方法将每个数据点作为可能的聚类中心，并根据各个数据点的周围的数据点密度来计算该点作为聚类中心的可能性。被选为聚类中心的数据点周期具有最高的数据点密度，同时该数据点附近的数据点被排除作为聚类中心的可能性；在选出第一个聚类中心后，从剩余的可能作为聚类中心的数据点中，继续采用类似的方法选择下一个聚类中心。这一过程一直持续到所有剩余的数据点作为聚为中心的可能性低于某一阈值为止。在 MATLAB 模糊逻辑工具箱中，提供了 subclust 函数来完成减法聚类的功能，该函数调用格式如下：

```
[C,S] = subclust(X,radii,xBounds,options)
```

式中，输入矩阵 $\boldsymbol{X}$ 包含了用于聚类的数据，$\boldsymbol{X}$ 的每一行为一个数据点向量；向量 radii 用于在假定数据点位于一个单位超立方体的条件下，指定数据向量的每一维的聚类中心的影响范围，即 radii 每一维的数值大于均在 0～1 之间，通常的取值范围为 0.2～0.5，如果数据点的维数为 2，则 radii=[0.5 0.25]指定了第一维数据的聚类中心的影响范围为数据空间宽度的一半，而第二维数据的聚类中心的影响范围为数据空间宽度的 1/4；xBounds 为一个 $2\times N$ 的矩阵，其中 N 为数据的维数（该矩阵用于指定如何将 $\boldsymbol{X}$ 中的数据映射到一个单位超立方体中，其第一行和第二行分别包括了每一维数据被映射到单位超立方体的最小和最大取值。例如，xBounds=[-10 -5;10 5]指定了第一位数据在[-10 10]之间的取值将被映射到[0 1]区间中；而第二维数据相应的区间范围为[-5 5]。如果 xBounds 为空或者没有指定 xBounds，则默认的映射范围为所有数据点的最小取值和最大取值构成的区间）；参数向量 options 用于指定聚类算法的有关参数（其定义如下：options(1)=squshFactor 为 squshFactor 用于与聚类中心的影

响范围 radii 相乘来决定某一聚类中心邻近的哪些数据点被排除作为聚类中心的可能性，默认值为 1.25；options(2)=acceptRatio 为 acceptRatio 用于指定在选出第一个聚类中心后，只有某个数据点作为聚类中心的可能性高于第一个聚为中心可能性值的一定比例（由 acceptRatio 的大小决定）才能被作为新的聚类中心的可能性，默认值为 0.5；options(3)=rejectRatio 为 rejectRatio 用于指定在选出第一个聚类中心后，只有某个数据点作为聚类中心的可能性值低于第一个聚类中心可能性值的一定比例（由 rejectRatio 的大小决定）才能被排除作为聚类中心的可能性，其默认值为 0.15；options(4)=verbose 为如果 verbose 为非零值，则聚类过程的有关信息将显示到窗口中，其默认值为 0)；函数的返回值 C 为聚类中心向量；向量 S 包含了数据点的每一维聚类中心的影响范围。



例如：

[C, S]=subclust(x, 0.5)命令表明坐标上聚类中心的影响范围都是数据空间宽度的 0.5，其余都采用默认值。

[C, S]=subclust(x, [0.5 0.25 0.3],[],[2.0 0.8 0.7])表明在 3 个坐标上（假设 X 是三维的坐标数据），聚类中心的影响范围分别为 0.5、0.25 和 0.3，采用默认的尺度变换，options 参数分别为 2.0、0.7 和 0。

【例 8-2】 利用模糊减法聚类方法将一组随机给定的二维数据分为两类。

其实现的 MATLAB 程序代码如下：

```
>> data = rand(100, 2);
plot(data(:,1),data(:,2),'p');
radii=0.3;
[C,S]=subclust(data,radii);
N=length(C);
hold on;
for i=1:N;
    plot(C(i,1),C(i,2),'mp','markersize',12,'Linewidth',1.5);
end
title('模糊减法聚类 (radii=0.3)');
```

运行程序，输出效果如图 8-13 所示。

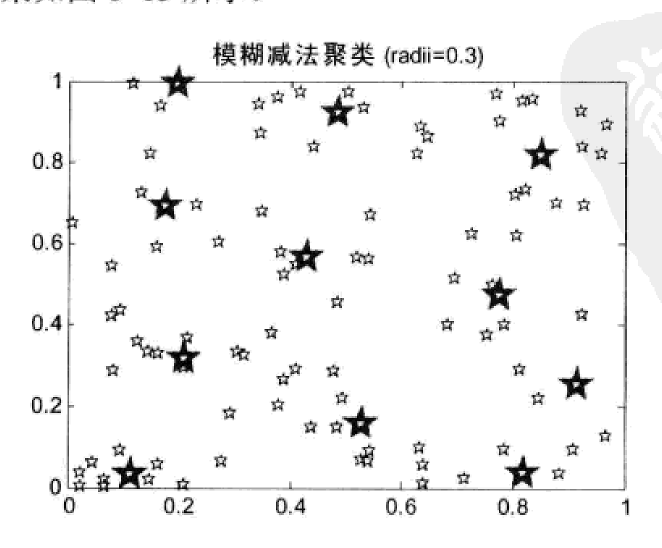


图 8-13 模糊减法聚类结果

8.3.3 基于减法聚类的模糊推理系统建模函数

在减法聚类的基础上可以进行模糊推理系统的建立，模糊逻辑工具箱提供的 genfis2 函数能够实现这一功能。genfi2 函数是一种快速的单次算法，不同于那些基于迭代过程的算法，它并不进行反复的优化过程。其调用格式如下：

```
fismat = genfis2(Xin,Xout,radii)
fismat = genfis2(Xin,Xout,radii,xBounds)
fismat = genfis2(Xin,Xout,radii,xBounds,options)
```

式中，Xin 和 Xout 为给定的输入数据和输出数据；参数 radii、xBounds 和 options 分别为减法聚类的参数，其详细说明参见减法聚类函数 subclust；输出参数 fismat 为 Takagi-Sugeno 型模糊推理系统的矩阵。该函数首先调用减法聚类函数 subclust 对输入、输出数据进行减法聚类，以决定输出和输入语言变量的隶属函数个数和规则个数，以及如何采用最小方差估计得到 Takagi-Sugeno 型推理规则结论部分的参数。

【例 8-3】 genfis2 函数示例。

```
>>  Xin1 =   7*rand(50,1);
        Xin2 = 20*rand(50,1)-10;
        Xin   = [Xin1 Xin2];
        Xout =   5*rand(50,1);
        fis = genfis2(Xin,Xout,0.5)
```

运行程序，输出结果如下：

```
fis =
             name: 'sug21'
             type: 'sugeno'
        andMethod: 'prod'
         orMethod: 'probor'
        impMethod: 'prod'
        aggMethod: 'max'
     defuzzMethod: 'wtaver'
            input: [1x2 struct]
           output: [1x1 struct]
             rule: [1x8 struct]
```

8.3.4 模糊 C-均值和减法聚类的图形用户界面

MATLAB 的模糊逻辑工具箱也提供了聚类的用户界面工具函数 subclust 和 fcm 及它们的所有参数选项。在 MATLAB 命令窗口中输入命令“findcluster”，便可打开如图 8-14 所示的模糊聚类图形用户界面。

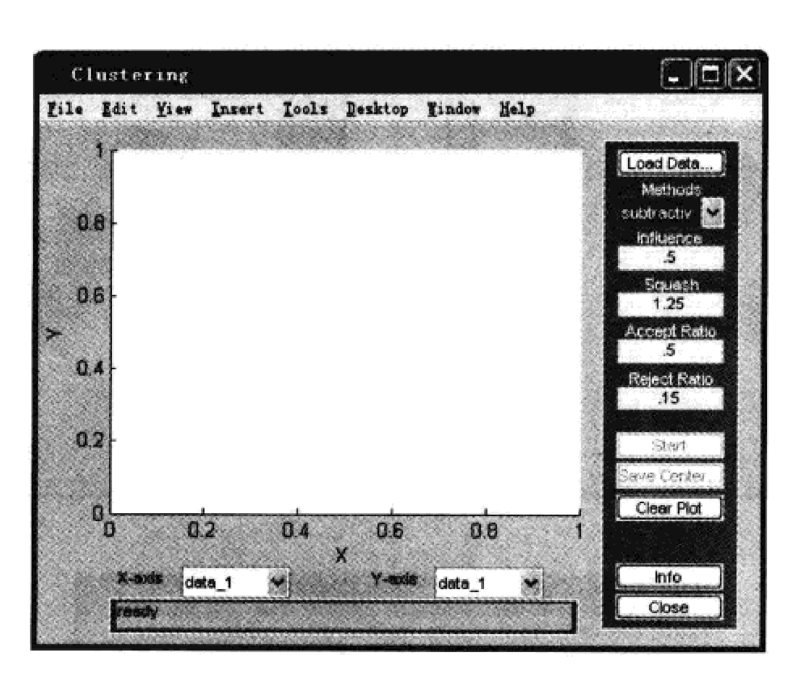



图 8-14　模糊聚类图形用户界面



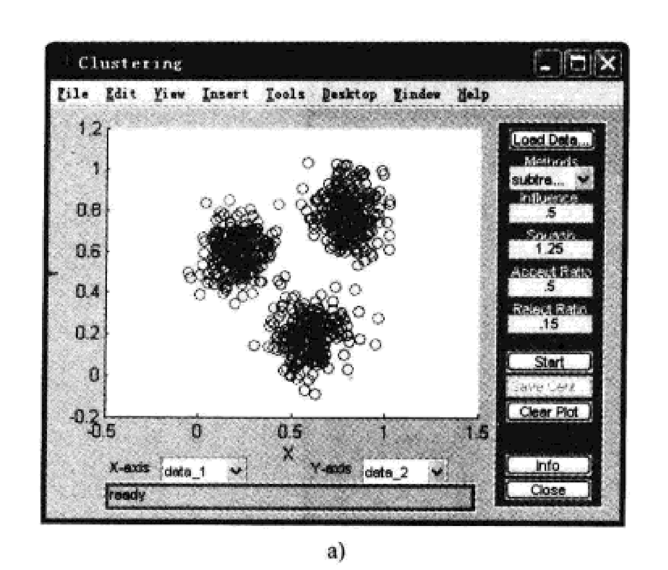
在图 8-14 所示的窗口中，右上方的按钮“Load Data...”可以选择具有.dat 扩展名的数据文件进行装载数据操作；Method 下拉列表框，可以选择使用模糊-C 均值聚类（fcm）或减法聚类（subclustiv）算法；其他选择框的功能随着 Method 算法选择 fcm 或 subclustiv 而发生相应的变化，这些主要涉及两种算法中的一些具体参数，如选择 subclustiv 算法，则界面上的参数 Influences、Squash、Accept Ratio 和 Reject Ratio 分别对应于函数 subclust 的 radii、squshFactor、acceptRatio 和 rejectRatio；而对于选择 fcm 算法，界面上的参数 Cluster Num、Max Iteration #、Min.Improvation 和 Exponent 分别对应于函数 fcm 的 options(1)、options(2)、options(3)和 options(4)。

右下方的“Start”按钮可以开始进行聚类计算，结果显示在 X-Y 二维的绘图区。对于多维的装载数据，可以通过绘图区下方的文本框来分别选择多维数据中的某个或两个方向作为 X 轴和 Y 轴的数据来显示；按钮“Save Center...”用于保存聚类中心文件；“Clear Plot”按钮用于清除当前显示在绘图区中的图形。

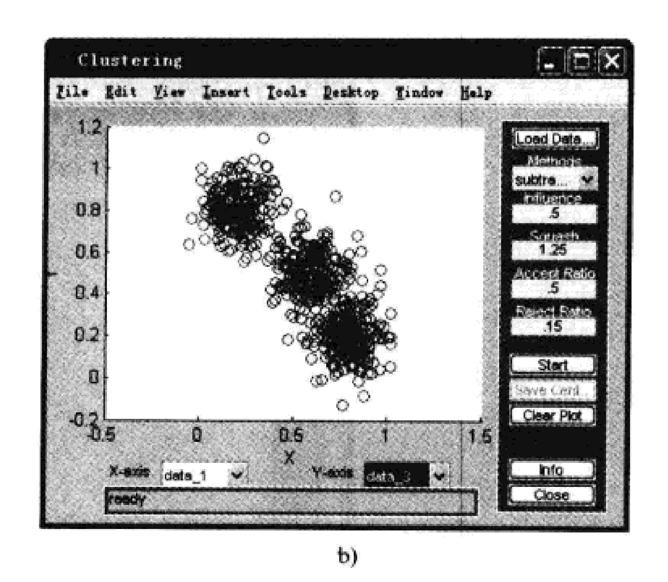
利用以下命令，模糊聚类图形窗口可自动加载 MATLAB 自带的一个数据文件“clusterdemo.dat”。

```
>> findcluster('clusterdemo.dat')
```

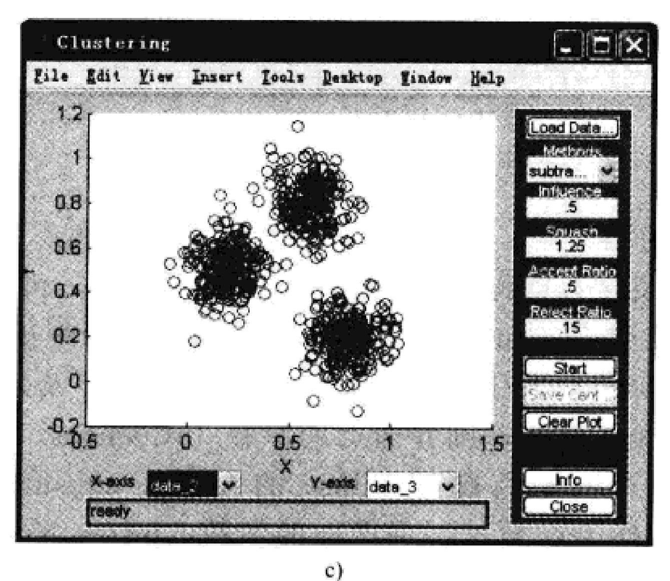
以下命令执行后，便可得到一个加载数据的模糊聚类图形窗口，通过窗口中的“Start”按钮进行聚类计算后，可以很快找到 3 个聚类中心，如图 8-15a 显示了数据文件 clusterdemo.dat 中数据 data_1 和 data_2 的关系，图 8-15b 显示了数据文件 clusterdemo.dat 中数据 data_1 和 data_3 的关系，图 8-15c 显示了数据文件 clusterdemo.dat 中数据 data_2 和 data_3 的关系。



a)



b)



c)

图 8-15　模糊聚类图形窗口

a）数据 data_1 和 data_2 的关系　b）数据 data_1 和 data_3 的关系　c）数据 data_2 和 data_3 的关系

【例 8-4】 利用模糊 C-均值和减法聚类的的图形用户将一组随机给定的二维数据分为两类。

（1）建立数据文件

利用以下 MATLAB 语句，在 MATLAB 工作空间产生一组随机给定的二维数据矩阵 X，并将其以 ASCII 码的形式保存到 MATLAB 当前工作目录的硬盘文件 xiudata.dat 中，以便于数据以磁盘文件（disk）的方式加载。

```
>> X=rand(123,3);
>> save xiudata.dat X -ascii
```

（2）启动模糊 C-均值和减法聚类的图形用户界面，并加载数据。

在 MATLAB 命令窗口中输入“findcluster”，便可打开图 8-14 所示的模糊聚类图形。在图 8-14 所示的窗口中，利用右上方的按钮“Load Data...”装载以上建立的数据文件

“xiudata.dat”中的数据，如图 8-16 所示。

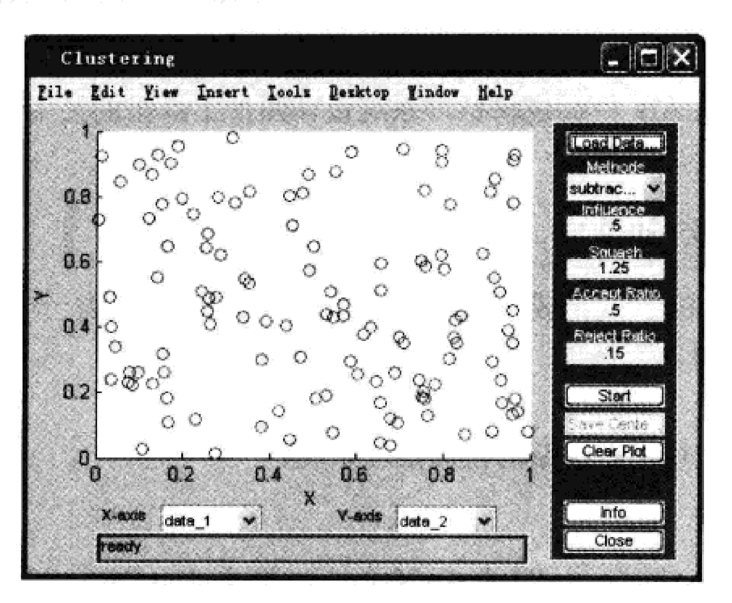
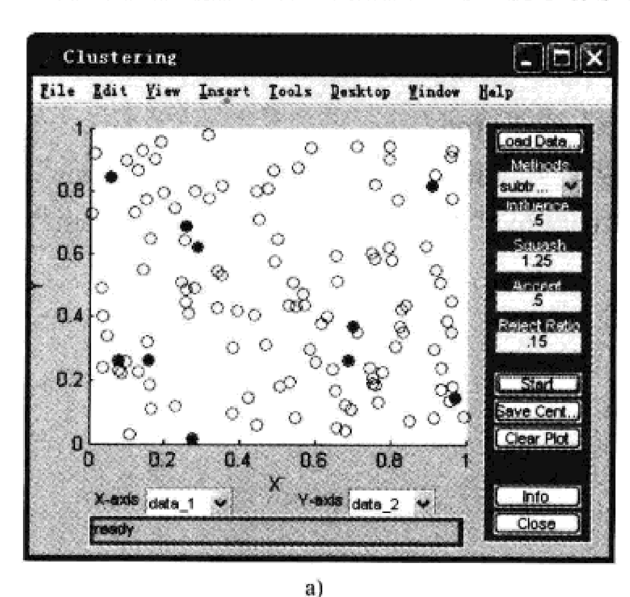


图 8-16 模糊聚类数据窗口

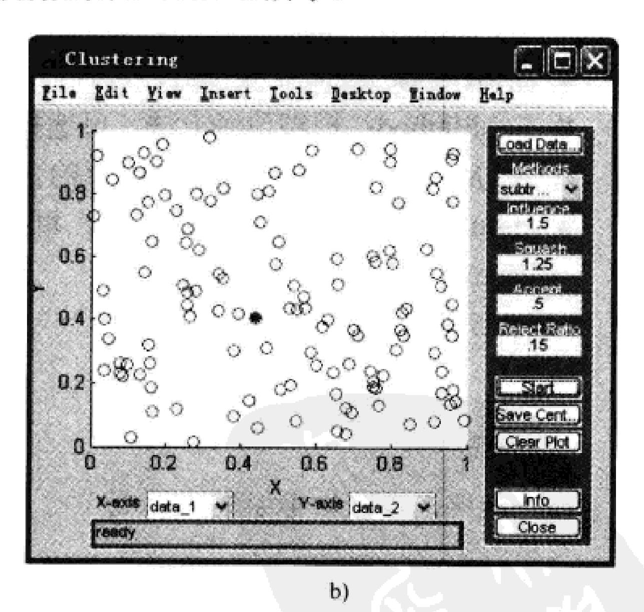
（3）聚类计算

在“Method”下拉列表框中选择使用减法聚类（Subclustiv）算法；参数 Influences 分别对为 0.5 和 1.5；其他下拉列表框的参数默认。

利用“Start”按钮可以开始进行聚类计算，结果显示在 X-Y 二维的绘图区中。对于应于 Influences=0.5 和 Influences=1.5 时的结果，分别如图 8-17a、b 所示。



a)



b)



图 8-17 模糊聚类结果

a)（Influences）radii=0.5 时的聚类效果 b)（Influences）radii=1.5 时的聚类效果

在图 8-17 所示的模糊聚类结果中，模糊聚类的中心用实心黑圆表示。当 radii=0.5 时，产生了 10 个模糊聚类中心，如图 8-17a 所示；当 raddii=1.5 时，仅产生了一个模糊聚类中心，如图 8-17b 所示。

8.4 模糊控制的相关函数

前面介绍了 MATLAB 模糊逻辑推理系统设计图形工具箱的使用方法，下面将介绍模糊逻辑推理系统的命令设计方法。

8.4.1 模糊推理系统的建立、修改与管理存储相关函数

MATLAB 以文件的形式把模糊逻辑系统的各个部分保存为一个整体，文件扩展名为“.fis”。同时可以对文件进行编辑、修改、管理和存储等作用。

1. newfis 函数

功能：newfis 函数用来创建新的模糊推理系统。

其调用格式如下：

```
a=newfis(fisName,fisType,andMethod,orMethod,impMethod, aggMethod,defuzzMethod)
```

其参数说明如下：

- a：为新创建的模糊推理系统在工作空间以矩阵的形式保存的文件名称，可依据 MATLAB 语法规则进行设定。该名称可用来引用模糊推理系统的属性值，如“a.input”、“a.andMetho”和“a.defuzzMethod”等。
- fisName：模糊推理系统的名称。
- fisType：模糊推理类型，可选择“Mamdani”或“Sugeno”。
- andMethod：与运算操作符，可选择“min”或“max”。
- orMethod：或运算操作符，可选择“max”或“probor”。
- impMethod：模糊蕴涵方法，可选择“min”或“prod”。
- aggMethod：各条规则推理结果的合成方法，可选择“max”、“sum”或“probor”。
- defuzzMethod：解模糊化方法，可选择“centriod”、“bisector”、“mom”、“lom”或“som”。

【例 8-5】 newfis 函数示例。

```
>> a=newfis('newsys');
getfis(a)
```

运行程序，输出结果如下：

```
>> a=newfis('newsys');
getfis(a)
        Name         = newsys
        Type         = mamdani
        NumInputs = 0
        InLabels   =
        NumOutputs = 0
        OutLabels =
        NumRules = 0
        AndMethod = min
```

```
        OrMethod = max
        ImpMethod = min
        AggMethod = max
        DefuzzMethod = centroid
ans =     newsys
```



从上述显示结果可以看出与设计要求完全一致，可用下述命令对模糊推理系统进行修改：

```
>> a.name='xiu';                    %修改系统模型名为“xiu”
a.defuzzMethod='bisector';          %修改解模糊化方法为“bisector”
a.aggemthod='proobr';               %修改规则合成算法为“proobr”;
a.impmethod='prod';                 %修改规则的蕴涵方法为“prod”;
getfis(a)                           %显示模糊逻辑系统属性
```

根据上述命令，在 Command Window 得到如下显示结果，从中可以看出模糊推理系统属性值设定要求已被修改。

```
        Name       = xiu
        Type       = mamdani
        NumInputs = 0
        InLabels  =
        NumOutputs = 0
        OutLabels =
        NumRules = 0
        AndMethod = min
        OrMethod = max
        ImpMethod = min
        AggMethod = max
        DefuzzMethod = bisector
ans =     xiu
```

2. writefis 函数

功能：将内存空间中以矩阵形式保存的模糊推理系统数据保存在磁盘上。

其调用格式如下：

```
writefis(fismat)
writefis(fismat,'filename')
writefis(fismat,'filename','dialog')
```

其参数说明如下：

- fismat：内存空间中的以矩阵格式保存的模糊推理系统名称。
- filename：磁盘上已存在的模糊推理系统名称。
- dialog：打开一个默认下模糊推理对话框类型。

使用第一个调用格式，将打开一个对话框，用于指定保存模糊推理系统的名称和位置。使用第二种格式，直接将内存空间中的模糊推理系统指定的保存到磁盘上已存在的文件

“filename”中；第三种调用格式，指定对话框类型。

【例 8-6】 writefis 函数示例。

```
>> a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
```

运行程序，输出结果如下：

```
ans =          my_file
```

3. readfis 函数

功能：从磁盘中读取模糊推理系统。

其调用格式如下：

```
readfis('filenme')
```

其为打开一个有 filename 指定的模糊推理系统的数据文件 filename.fis，并将其加载到当前的工作空间中。

【例 8-7】 readfis 函数示例。

```
>> a = readfis('tipper');
plotfis(a)
```

运行程序，输出效果如图 8-18 所示。

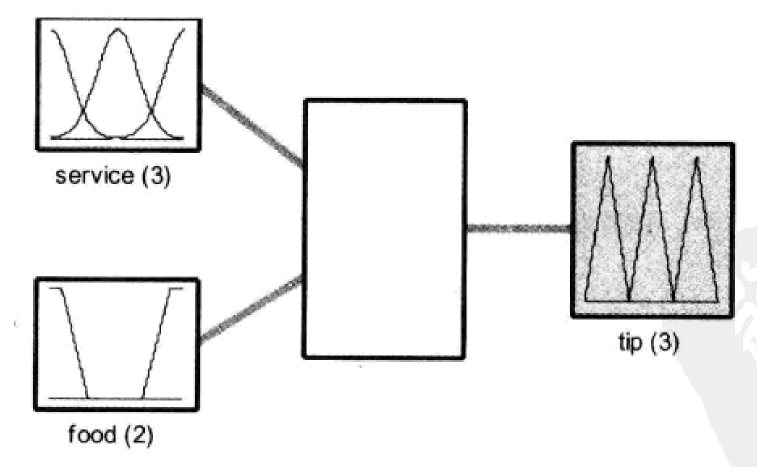


图 8-18 tipper 关系图

4. getfis 函数

功能：获取模糊推理系统的属性。

其调用格式如下：

```
getfis(a)
getfis(a,'fisprop')
getfis(a,'vartype',varindex,'varprop')
```

其参数说明如下：



- a：内存空间模糊推理模型名称。
- fisprop：用于指定期望获取的某一属性。此时 getfis 函数仅输出指定的属性值。
- vartype：变量类型，varindex 为变量索引，varprop 为变量属性。

【例 8-8】 getfis 函数示例。

```
>> a = readfis('tipper');
getfis(a)
```

运行程序，输出结果如下：

```
        Name        = tipper
        Type        = mamdani
        NumInputs = 2
        InLabels    =
              service
              food
        NumOutputs = 1
        OutLabels =
              tip
        NumRules = 3
        AndMethod = min
        OrMethod = max
        ImpMethod = min
        AggMethod = max
        DefuzzMethod = centroid
ans =   tipper
```

输入以下命令：

```
>> getfis(a,' DefuzzMethod ')
```

运行程序，输出结果如下：

```
ans =   centroid
```

输入以下命令：

```
> getfis(a,'input',2)
```

运行程序，输出结果如下：

```
        Name =       food

        NumMFs =     2
        MFLabels =
```

```
                rancid
                delicious
        Range =      [0 10]
ans =
        Name: 'food'
     NumMFs: 2
         mf1: 'rancid'
         mf2: 'delicious'
      range: [0 10]
```

输入以下命令：

```
>> getfis(a,'input',2,'mf',2)
        Name = delicious
        Type = trapmf
        Params = [7 9 10 10]
```

运行程序，输出结果如下：

```
ans =
        Name: 'delicious'
        Type: 'trapmf'
     params: [7 9 10 10]
```

5．showfis 函数

功能：以分行的形式显示模糊推理系统矩阵的所有属性。

其调用格式如下：

```
showfis(fismat)
```

式中，fismat 为内存空间中以矩阵格式保存的模糊推理系统名称。

【例 8-9】 showfis 函数示例。

```
>> a = readfis('tipper');
```

运行程序，输出结果如下：

```
showfis(a)
1.  Name              tipper
2.  Type              mamdani
3.  Inputs/Outputs    [2 1]
4.  NumInputMFs       [3 2]
5.  NumOutputMFs      3
6.  NumRules          3
7.  AndMethod         min
8.  OrMethod          max
9.  ImpMethod         min
10. AggMethod         max
11. DefuzzMethod      centroid
```

```
12. InLabels         service
13.                   food
14. OutLabels        tip
15. InRange          [0 10]
16.                   [0 10]
17. OutRange         [0 30]
18. InMFLabels       poor
19.                   good
20.                   excellent
21.                   rancid
22.                   delicious
23. OutMFLabels      cheap
24.                   average
25.                   generous
26. InMFTypes        gaussmf
27.                   gaussmf
28.                   gaussmf
29.                   trapmf
30.                   trapmf
31. OutMFTypes       trimf
32.                   trimf
33.                   trimf
34. InMFParams       [1.5 0 0 0]
35.                   [1.5 5 0 0]
36.                   [1.5 10 0 0]
37.                   [0 0 1 3]
38.                   [7 9 10 10]
39. OutMFParams      [0 5 10 0]
40.                   [10 15 20 0]
41.                   [20 25 30 0]
42. Rule Antecedent  [1 1]
43.                   [2 0]
44.                   [3 2]
42. Rule Consequent  1
43.                   2
44.                   3
42. Rule Weight      1
43.                   1
44.                   1
42. Rule Connection  2
43.                   1
44.                   2
```



6. setfis 函数

功能：设置模糊推理系统的属性。

其调用格式如下：



```
a = setfis(a,'fispropname','newfisprop')
a = setfis(a,'vartype',varindex,' fispropname ','newvarprop')
a = setfis(a,'vartype',varindex,'mf',mfindex, ' fispropname ','newmfprop');
```

其参数说明如下：

- fispropname：模糊推理系统 a 属性的名称。
- newfisprop：期望设置的新属性值。
- vartype：变量类型。
- varindex：变量索引号。
- mf：隶属函数。
- mfindex：隶属函数的索引号。

可设置的属性值有：

- name：模糊推理系统的名称。
- type：模糊推理系统的类型。
- Inputs：模糊推理系统的输入变量个数。
- Outputs：模糊推理系统的输出变量个数。
- NumInputMFs：输入隶属函数个数。
- NumOutputMFs：输出隶属函数个数。
- NumRules：模糊推理系统的规则个数。
- Andmethod：模糊推理系统的与运算方法。
- Ormethod：模糊推理系统的或运算方法。
- Impmethod：模糊推理系统的模糊蕴涵方法。
- AggMethod：模糊推理系统的各个规则合成方法。
- DefuzzMethod：模糊推理系统的解模糊化方法。

【例 8-10】 setfis 函数示例。

```
>> a=newfis('xiusys')
a1=setfis(a,'andMethod','prod');
a1=addvar(a,'input','E',[-3,3]);
a=addmf(a,'input',1,'NB','zmf',[-3,2]);
a=addmf(a,'input',1,'NM','trimf',[-3,-2,-1])
```

运行程序，输出结果如下：

```
a =
              name: 'xiusys'
              type: 'mamdani'
         andMethod: 'min'
          orMethod: 'max'
      defuzzMethod: 'centroid'
         impMethod: 'min'
         aggMethod: 'max'
             input: []
            output: []
```



```
rule: []
```

输入以下命令：

```
>> getfis(a,'NumInputMFs')
```



运行程序，输出结果如下：

```
ans =
     []
```

8.4.2 模糊规则建立与修改相关函数

在 MATLAB 模糊逻辑工具箱中提供了相关模糊规则建立和操作的函数，下面作相应介绍。

1. addrule 函数

功能：向模糊逻辑推理系统添加模糊规则。

其调用格式如下：

```
a = addrule(A,ruleList)
```

式中，A 为模糊推理系统在工作空间中的矩阵名称；ruleList 以向量的形式给出需要添加的模糊规则。该向量的格式有严格的要求。如果模糊推理系统有 m 个输入模糊语言变量和 n 个输出模糊语言变量，则向量 ruleList 的列数必须为 m+n+2，而行数任意。在 ruleList 的每一行中，前 m 个表示数字表示各输入语言变量的语言值，其后的 n 个数字输出表示输出语言变量的语言值，第 m+n+1 个数字是该规则的权重，权重在[0 1]区间，一般设为 1。第 m+n+2 个数字为 0 或 1 两个值之一。如果取 1，则表示模糊规则前件的各语言变量是“与”的关系；如果取 0，则表示模糊规则前件的各语言变量是“或”的关系。

2. parsrule 函数

功能：解析模糊规则。

其调用格式如下：

```
fis2 = parsrule(fis,txtRuleList)
fis2 = parsrule(fis,txtRuleList,ruleFormat)
fis2 = parsrule(fis,txtRuleList,ruleFormat,lang)
```

函数 parsurle 对给定的模糊语言规则进行解析并添加到模糊推理系统矩阵中。其输入参数如下：

- fis2：规则解析后模糊推理系统矩阵。
- fis：规则解析前模糊推理系统矩阵。
- txtRuleList：以模糊语句表示的模糊语言规则。
- ruleFormat：模糊规则的格式，包括语言型（verbose）、符号型（symbolic）和索引型（indexed）。
- lang：模糊规则的长度。

【例 8-11】 parsrule 函数示例。

```
>> a = readfis('tipper');
ruleTxt = 'if service is poor then tip is generous';
a2 = parsrule(a,ruleTxt,'verbose');
showrule(a2)
```

运行程序，输出结果如下：

```
ans =
    1. If (service is poor) then (tip is generous) (1)
```

3. showrule 函数

功能：显示模糊规则。

其调用格式如下：

```
showrule(fis)
showrule(fis,indexList)
showrule(fis,indexList,format)
showrule(fis,indexList,format,Lang)
```

其参数说明如下：

- fis 为模糊推理系统的矩阵名称。
- indexList 为规则编号，规则编号可以用向量的形式指定多个规则。
- format 为模糊规则的显示格式，即语言型（verbose）、符号型（symbolic）和索引型（indexed）。
- lang 为显示模糊规则的长度。

【例 8-12】 showrule 函数示例。

```
>> a = readfis('tipper');
showrule(a,1)
ans =
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
>> showrule(a,2)
ans =
2. If (service is good) then (tip is average) (1)
>> showrule(a,[3 1],'symbolic')
ans =
3. (service==excellent) | (food==delicious) => (tip=generous) (1)
1. (service==poor) | (food==rancid) => (tip=cheap) (1)
>> showrule(a,1:3,'indexed')
ans =
1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2
```

8.4.3 模糊推理计算与解模糊化的相关函数

MATLAB 提供了模糊推理计算和解模糊化的函数分别为 evalfis 和 defuzz，以及生成的

模糊推理系统输出曲面显示 gensurf 函数。

1．evalfis 函数

功能：执行模糊推理计算。

其调用格式如下：



```
output= evalfis(input,fismat)
```

其参数说明为：该函数计算以 input 为输入向量的模糊推理系统的输出模糊向量 output。fismat 为模糊推理系统在工作空间中的矩阵名称。

【例 8-13】 evalfis 函数示例。

```
>> fismat = readfis('tipper');
out = evalfis([2 1; 4 9],fismat)
```

运行程序，输出结果如下：

```
out =
     7.0169
    19.6810
```

2．defuzz 函数

功能：执行模糊推理输出的解模糊化处理。

其调用格式如下：

```
out = defuzz(x,mf,type)
```

其参数说明如下：

- x：参数 x 是变量的论域范围。
- mf：为等解模糊化处理的模糊集合。
- type：是解模糊化的方法。解模糊化方法包括 centroid（重心法）、bisector（面积平方法），mom（平均最大隶属度法）、som（最大隶属度取最小值法）和 lom（最大隶属度取最大值法）。

【例 8-14】 defuzz 函数示例。

```
>> x = -10:0.1:10;
mf = trapmf(x,[-10 -8 -4 7]);
xx = defuzz(x,mf,'centroid')
```

运行程序，输出结果如下：

```
xx =
    -3.2857
```

输入以下命令：

```
>> x=-4:0.1:4;
mf1=trimf(x,[-3 -2 -1]);
mf2=trimf(x,[-1 0   1]);
mf=max(mf1,mf2);
```

```
y=defuzz(x,mf,'lom')
```

运行程序，输出结果如下：

```
y =
    -2
```

3. gensurf 函数

功能：生成模糊推理系统输出曲面并显示。

其调用格式如下：

```
gensurf(fis)
gensurf(fis,inputs,output)
gensurf(fis,inputs,output,grids)
gensurf(fis,inputs,output,grids,refinput)
```

其参数说明如下：

- 参数 fis 为模糊推理系统在工作空间的矩阵名称。
- inputs 为模糊推理系统的一个或两个输入语言变量。
- output 为模糊推理系统输出语言变量。
- grids 为用于指定 XY 坐标系的网格数目，当系统输入变量多于两个时。
- refinput 用于指定保持不变的输入量。

【例 8-15】 gensurf 函数示例。

```
>> a = readfis('tipper');
gensurf(a)
```

运行程序，输出效果如图 8-19 所示。

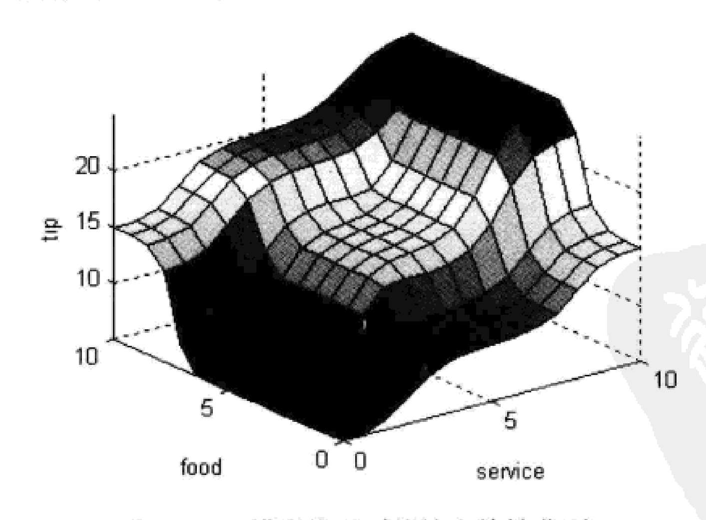


图 8-19 模糊推理系统输出特性曲面

8.5 模糊与 PID 控制器仿真设计

为了能有效地减小模糊控制中的稳态误差，在常规模糊控制器的基础上又派生了一些模糊控制器。本节将对其进行简单的讨论。



8.5.1 FIS 与 Simulink 的连接

模糊控制系统的设计与仿真，是以仿真模型图为基础的，仿真模型图由 Simulink 中的模块连接构成。在构成模糊控制系统仿真模型图时，必须用“Fuzzy Logic Toolbox”（模糊逻辑工具箱）中的“Fuzzy Logic Controller”（模糊逻辑控制器）模块，下面先对它进行介绍。

1．模糊逻辑工具箱介绍

进入 Simulink Library Browser 后，双击屏幕左侧子目录“Fuzzy Logic Toolbox”，单击鼠标右键，在弹出的菜单中选择“Open Fuzzy Logic Toolbox Library”命令，系统弹出如图 8-20 所示的模糊逻辑工具箱窗口。

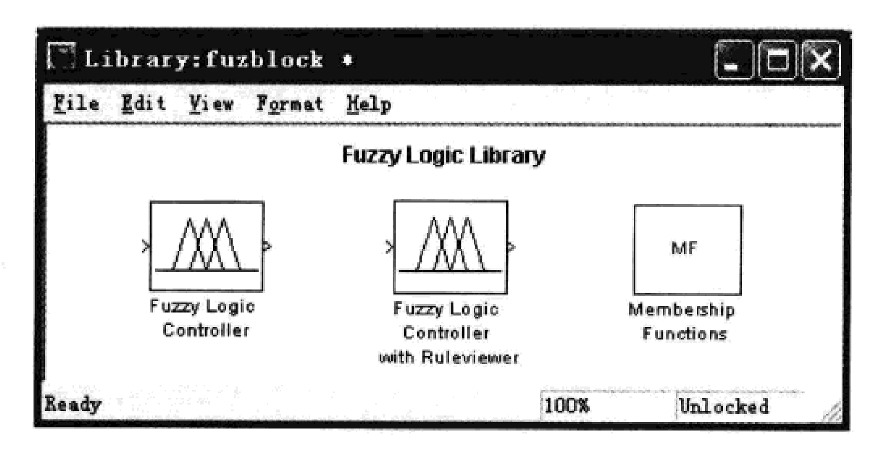


图 8-20 模糊逻辑工具箱窗口

模块“MF”（Membership Functions）备有模糊集合的各处隶属函数，可供选用。

模块“Fuzzy Logic Controller”和“Fuzzy Logci Cotroller with Ruleviewer”都是“模糊逻辑控制器”，它们的差异只是后者带有规则观测窗，在仿真时会显示出系统的“模糊规则观测窗”，可以观测到模糊推理的具体实施过程。

把“Fuzzy Logic Controller”模块拖入“模型编辑器”；再用鼠标右键单击，会弹出一个菜单；单击菜单中的“Look Under Mask”命令，就显现出它的内部结构，如图 8-21 所示。

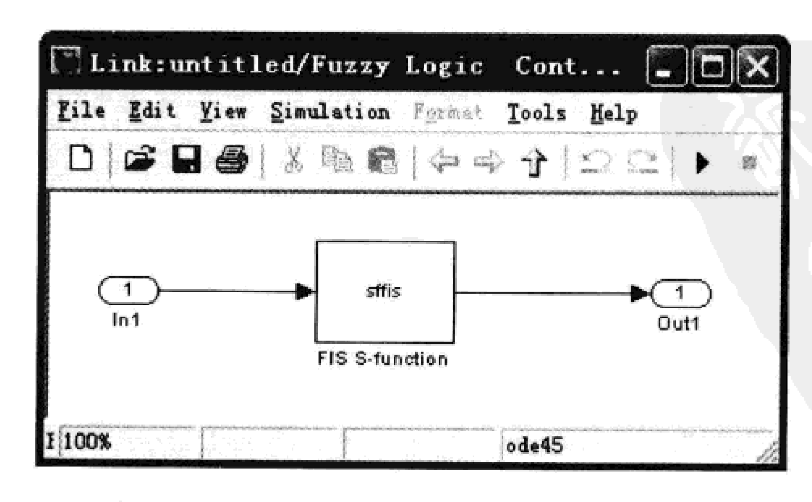


图 8-21 Fuzzy Logic Controller 内部结构子系统

同样对“Fuzzy Logic Controller with Ruleviewer”模块进行上述操作，则得出其内部

结构如图 8-22 所示，其上显示有模糊规则的动画（Animation）模块。用鼠标右键单击图 8-22 界面上的“Fuzzy Logic Controller”模块，再单击弹出菜单中的“Look Under Mask”命令，则得出一个与图 8-21 完全一样的结构图，其中也含有一个“FIS S-function”。两个模糊逻辑控制模块的 FIS S-Function 模框中，现在都写着“sffis”，表明都未嵌入 FIS 结构文件。

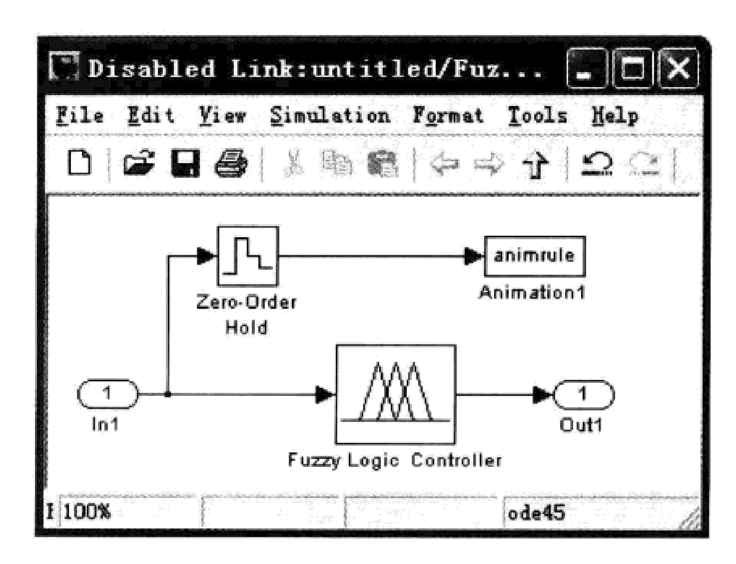


图 8-22　Fuzzy Logic Controller with Ruleviewer 内部结构子系统

2. 把 FIS 嵌入模糊逻辑控制器的方法

在 GUI 的 FIS 编辑器中编辑的 FIS 结构文件，可以按下述步骤嵌入“Fuzzy Logic controller”中。

（1）把 FIS 结构文件送入工作空间

MATLAB 的“工作空间”就像演出用的大舞台，仿真前必须把各个模块送入其中。把 FIS 结构文件送入工作空间有以下两种方法。

1）在 MATLAB 主窗口中用命令 readfis。

例如，要把编好并储存的 FIS 结构文件“xiunews”送入工作空间，就在 MATLAB 主窗口输入：

```
myorder=readfis（'xiunews'）
```

回车，得出该文件的结构列表：

```
myorder =
             name: ' xiunews '
             type: 'mamdani'
        andMethod: 'min'
         orMethod: 'min'
     defuzzMethod: 'centroid'
        impMethod: 'max'
        aggMethod: 'max'
            input: [1x2 struct]
           output: [1x3struct]
```



rule: [1x8 struct]

指令中赋值号“=”右边的“xiunews”可以换成其他名称。

于是在 FIS 编辑器中编辑的结构文件“xiu”，就被送入“工作空间”。

也可以输入“xiunews =readfis（' xiunews '）;”，由于句末有分号，回车不显示文件结构列表。



2）在 FIS 编辑器中使用鼠标。

- 在 MATLAB 主窗口中，输入“fuzzy xiunews”，得出“FIS Editor：xiunews”界面。
- 在该界面上，单击“File”菜单下的“Export”选项下的“To Workspace...”命令，系统弹出如图 8-23 所示的保存当前 FIS 到工作空间。

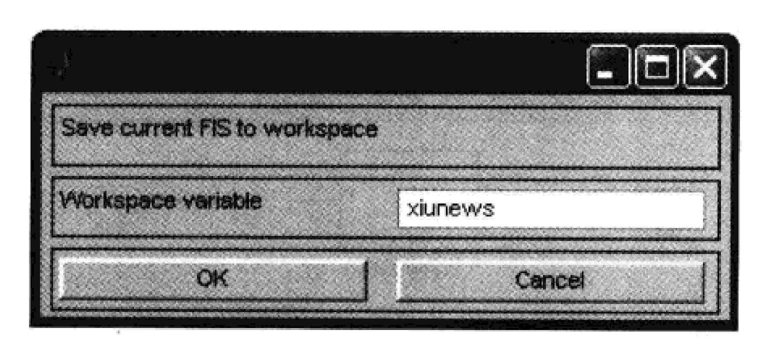


图 8-23 保存 FIS 到工作空间

- 在对话框界面上“Workspace variable”（工作空间变量）右侧的编辑框内，输入文件名称，此处为“xiunews”。
- 单击对话框中的功能按钮“OK”按钮，完成送入“工作空间”。

（2）把 FIS 结构文件嵌入 Fuzzy Logic Controller 模块

嵌入工作必须在“模型编辑器”中进行。

1）在编辑仿真模型图中，当用到“Fuzzy Logic Controller”模块时，首先从“Fuzzy Logic Toolbox”子库中把它拖入仿真模型编辑器界面上。

2）双击“模型编辑器”中的“Fuzzy Logic Controller”模块，系统弹出参数对话框，如图 8-24 所示。

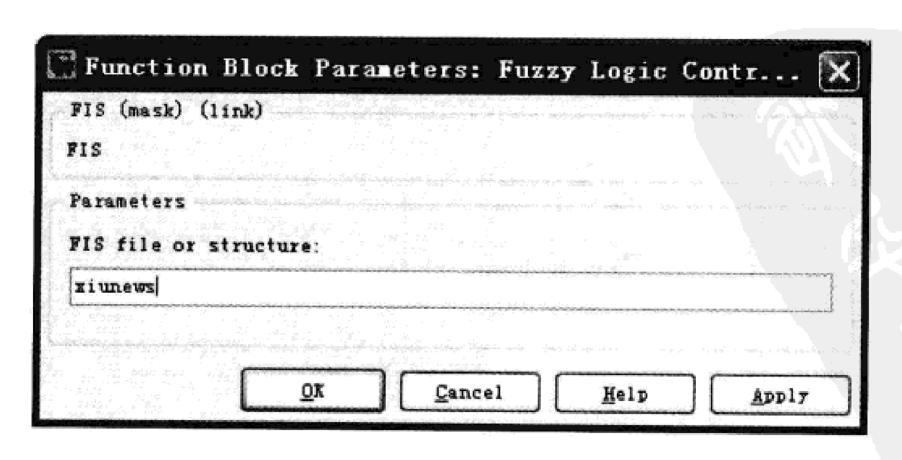


图 8-24 Fuzzy Logic Controller 的函数模框参数对话框

3）在“FIS file or structure：”下的编辑框内填入文件名“xiunews”，再单击界面下边的

"OK"功能按钮，即完成嵌入工作。

（3）嵌入成功性的检查

嵌入成功与否关系到模糊逻辑工具箱中的模块与Simulink是否连接成功，可否进行仿真的问题，仿真前必须进行检查。检查连接成功与否的方法是：用鼠标右键单击模型编辑器中的"Fuzzy Logic Controller"模块，在弹出的菜单中单击"Look Under Mask"选项，系统弹出"检查嵌入"对话框，如图8-25所示。

对话框中"FIS Wizard"模框内写有"FIS"，表明FIS嵌入成功，并且已和Simulink成功连接；如果写着"sffis"则表示连接失败，需要重新嵌入。

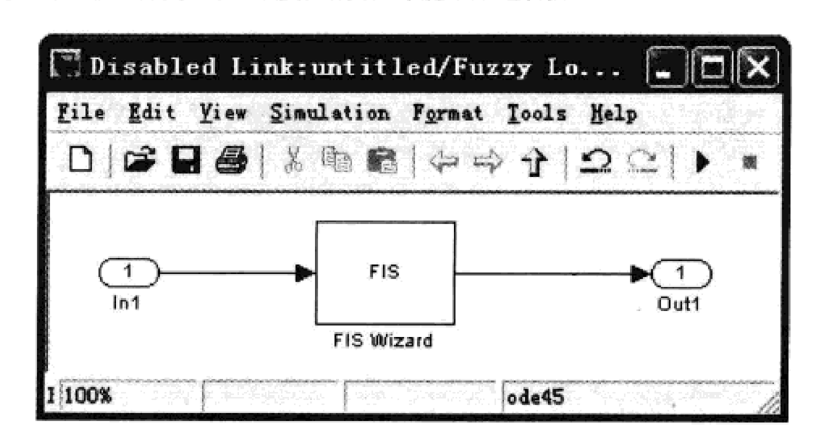


图8-25　检查嵌入对话框

重进行嵌入连接前，首先查看嵌入的结构文件（这里是"xiunews"）是否已经送入"工作空间"；其次再检查嵌入连接的每个步骤是否正确。

把FIS结构文件嵌入"Fuzzy Logic Controller"模块，就表明已经建立的FIS结构文件与Simulink实现了连接，这个调入仿真模型图的"Fuzzy Logic Controller"模块，已经可以与其他模块连接并进行仿真。

8.5.2 模糊-PI双模控制系统仿真设计

1. 模糊-PI双模控制系统结构

综合了模糊控制的动态性能及PI控制的稳态性能的模糊-PI双模控制系统框图如图8-26所示。其控制原理是，当系统偏差较大时，通过开关选择模糊控制器，以达到较好的动态特性；当系统偏差较小时，将开关切换到基本PI控制器，以获得较好的稳态性能。

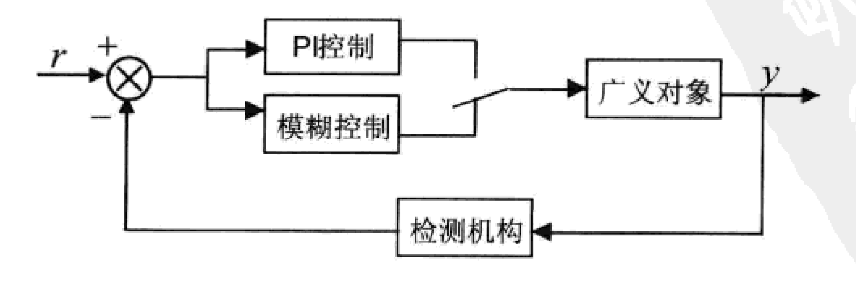


图8-26　模糊-PI双模控制系统结构

以广义对象为 $G(s)=\dfrac{10}{s(0.1s+1)(0.2s+1)}$ 的系统为例，试设计一个模糊-PI 双模控制系

统，并借助 MATLAB 模糊逻辑工具箱仿真分析。

2．控制器设计

（1）PI 控制器设计

为获得较好的稳态控制效果，首先假定在 PI 控制器单独作用状态下设计 PI 控制器。设计 PI 控制器的 MATLAB 程序代码如下，可获得初步的 PI 控制器结构参数。



```
>> num=10;
den=conv([1 0],conv([0.1 1],[0.2,1]));
G=tf(num,den);
s=tf('s');
[Gm,Pm,Wcg,Wcp]=margin(G);          %计算频域响应,增益标量 Gm 和剪切频率 Wcp
Tc=2*pi/Wcg;                        %计算剪切频率对应的时间周期 Tc
%PI 控制器
PIKp=0.4*Gm;                        %频率响应整定法算 PI 控制器
PITi=0.8*Tc;
PIGc=PIKp*(1+1/(PITi*s));
Gk=PIGc*G;
sys=feedback(Gk,1,-1);
step(sys,'r-');                     %绘制闭环阶跃响应曲线
title('PI 控制单位阶跃响应')
xlabel('时间');ylabel('幅值');
PIKp, PITi                          %在窗口显示 Kp 和 Ti 系数
```

运行以下程序，可得到整定后系统在 PI 控制作用下控制器参数 Kp=0.6，T_i=0.88，单位阶跃响应曲线如图 8-27 所示。

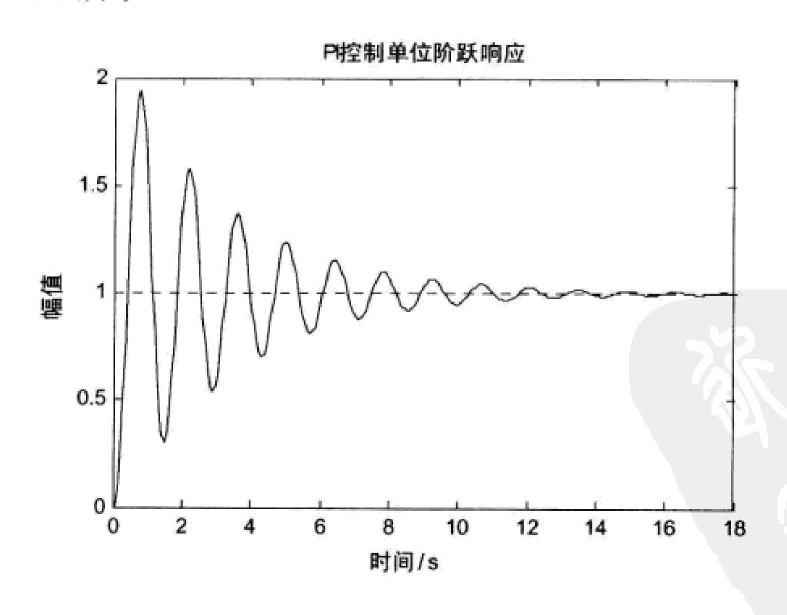


图 8-27　初始整定后 PI 控制闭环系统单位阶跃响应曲线

从图 8-27 初始整定后 PI 控制单位阶跃响应曲线可以看出，系统稳态性能并不理想，因此需要根据 PI 控制器参数对控制性能的影响对 Kp 和 T_i 进行修正。修正后闭环系统单位阶跃响应曲线如图 8-28 所示，参数 Kp=0.4，T_i=10。

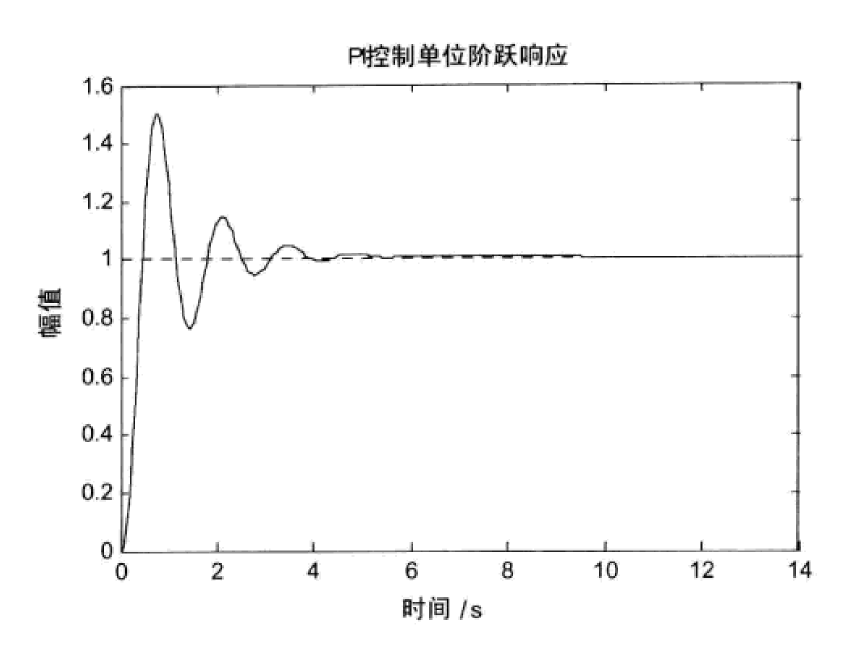


图 8-28　修正后闭环系统单位阶跃响应曲线

（2）模糊控制器设计

1）模糊控制器结构：根据给定要求，模糊控制器采用二维模糊控制器，其结构如图 8-29 所示。模糊控制器输入偏差 e 为给定输入信号与反馈信号之差，即 $e=r-y$。输入 ec 为偏差的变化率 $ec=\mathrm{d}e/\mathrm{d}t$。输出 u 为控制量。K_e、K_{ec}、K_u 分别为偏差 e、偏差变化率 ec 及控制量 u 的量化因子。

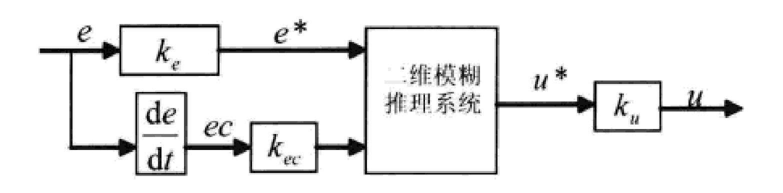


图 8-29　二维模糊控制器框图

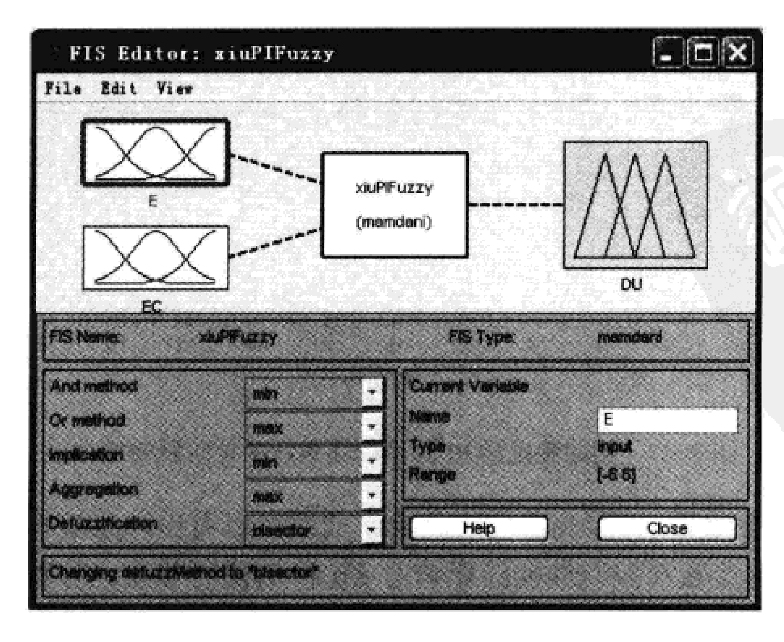


图 8-30　模糊控制器变量及高层属性编辑器

2）模糊语言变量设计：设二维模糊推理输入模糊语言变量为 e 和 ec，模糊论域为[-6 6]，输出模糊语言变量为 U，模糊论域为[0 10]。实际的偏差为 e，在单位阶跃响应信号作用下，其基本论域设定为[-0.5 0.5]。实际的偏差变化率信号 ec 的基本论域为[-1 1]。实际输出控制量 u 的基本论域设定为[0 10]。因此可确定偏差的量化因子 K_e=12，偏差变化率的量化因子 K_{ec}=6。输出量 u 的量化因子 K_u=1。具体数值在 Simulink 仿真模型中可进行修改。将模糊语言变量 E 的语言值设定为 6 个，即，即{负大（NB），负中（NM），负小{NS}，正小（PS），正中（PM），正大（PB）}；将偏差变化率的模糊语言变量 ec 的语言值设定为 5 个，即{负大（NB），负小（NS），零（Zero），正小（PS），正大（PB）}；将输出模糊语言变量 U 的语言值设定为 5 个，即{零（Zero），正微（PW），正大（PS），正中（PM），正大（PB）}。在 Command Window 输入“fuzzy”命令，打开模糊逻辑推理系统编辑器，并设定模糊语言变量及其高层属性，如图 8-30 所示。设定输入/输出模糊语言变量的隶属函数如图 8-31～图 8-33 所示。



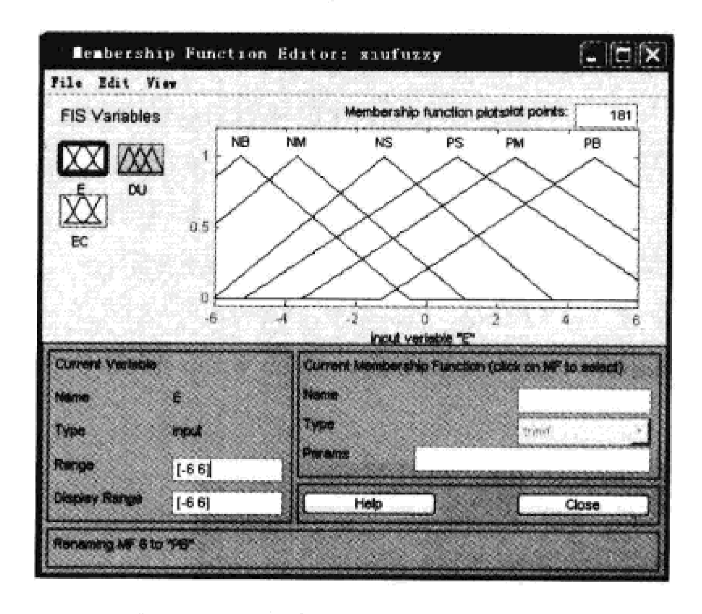


图 8-31　偏差 e 的隶属函数曲线

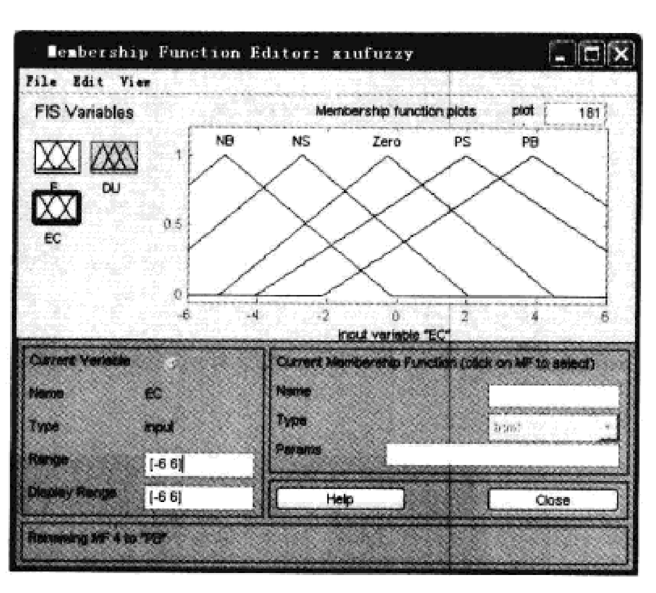


图 8-32　偏差变化率 ec 的隶属函数曲线

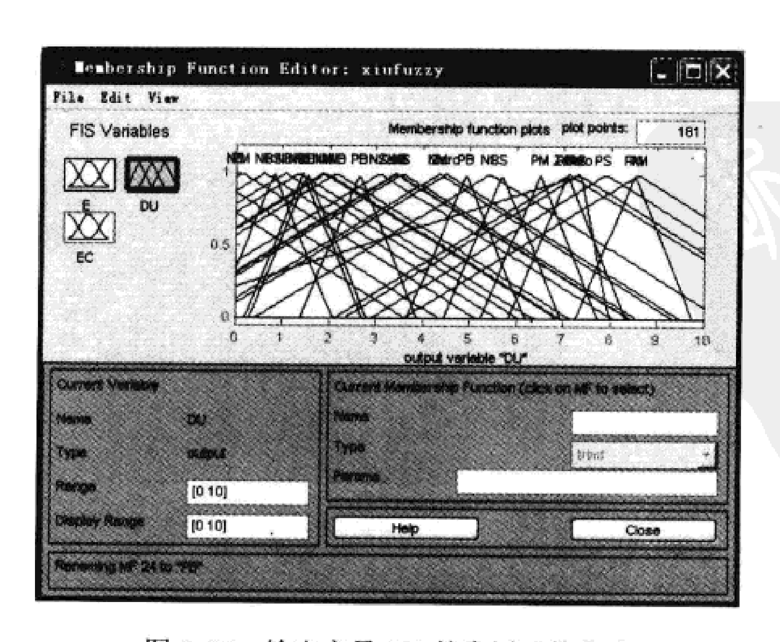


图 8-33　输出变量 DU 的隶属函数曲线

3）模糊规则设计。模糊-PI 双模控制中的模糊控制器主要在工作过渡过程，因此希望模糊控制能加快系统响应速度，再根据自动控制基本理论，设计模糊规则见表 8-1。

表 8-1　模糊-PI 双模控制器的模糊控制规则表

DU		E					
		NB	NM	NS	PS	PM	PB
EC	NB	NB	NB	NM	NS	Zero	PS
	NS	NB	NB	NS	Zero	PS	PM
	Zero	NB	NM	NS	PS	PM	PB
	PS	NM	NS	Zero	PM	PB	PB
	PB	NS	Zero	PS	PM	PB	PB

规则解释：以规则“IF E=PB and EC=Zero then U=PB”为例，若偏差为正大（PB）且偏差变化率为零，即被控量反馈值远小于设定值而且没有减小的趋势，为了获得较快的响应速度，应该加大控制量，即 U=PB。再以规则“IF E=NB and EC=NB then U=NB”为例说明，当偏差为 NB（“负大”）时，且偏差变化率也为“负大”，即被控量超调很大，而且超调继续加大，为了使被控量快速返回稳态值，应以最大值来减小控制量。单击“Edit”菜单下的“Rules”命令，打开规则编辑器，编辑表 8-1 所示的模糊规则，结果如图 8-34 所示。再分别单击“View”菜单下的“Rules”命令和“View”菜单下的“Surface”命令，打开模糊规则观察器和模糊控制输入输出关系曲面，如图 8-35 和图 8-36 所示。

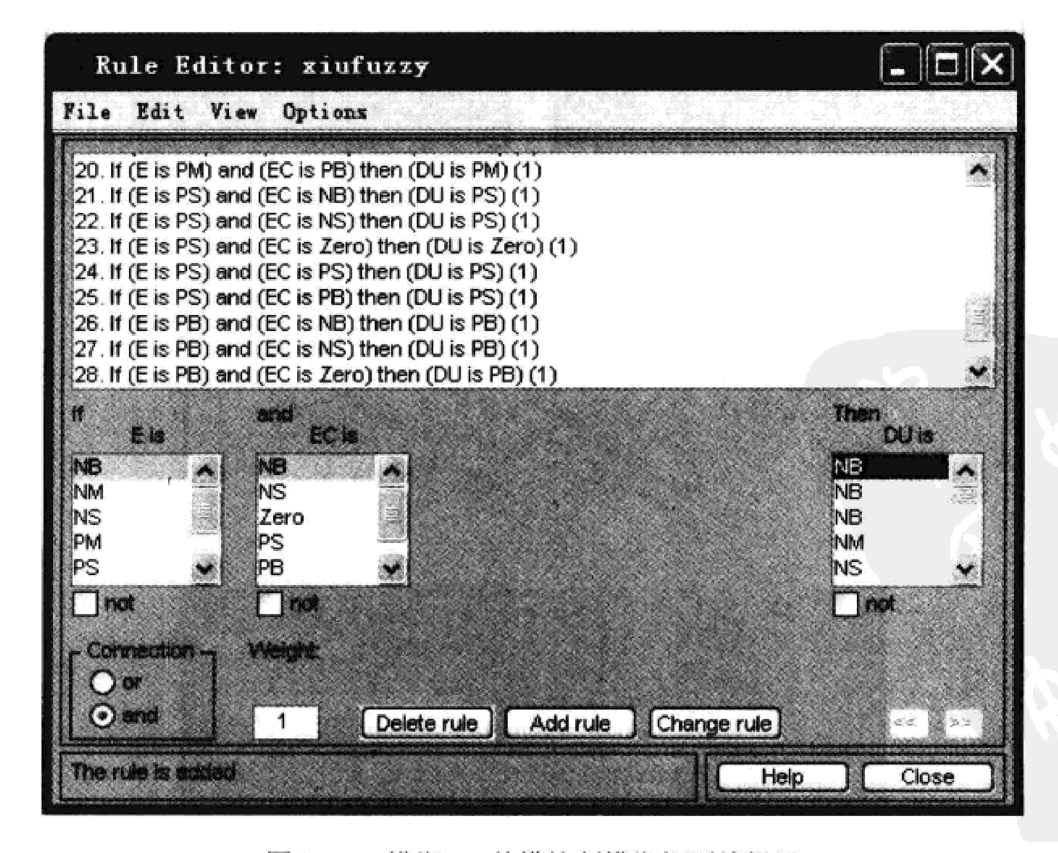


图 8-34　模糊-PI 单模控制模糊规则编辑器

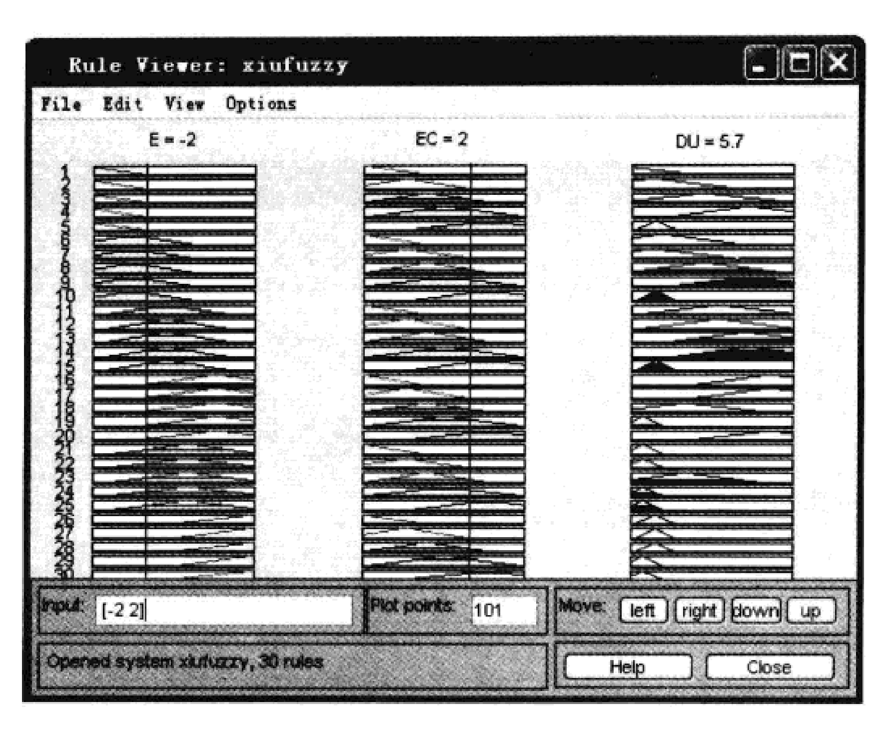


图 8-35 模糊-PI 双模控制模糊推理控制规则观察器

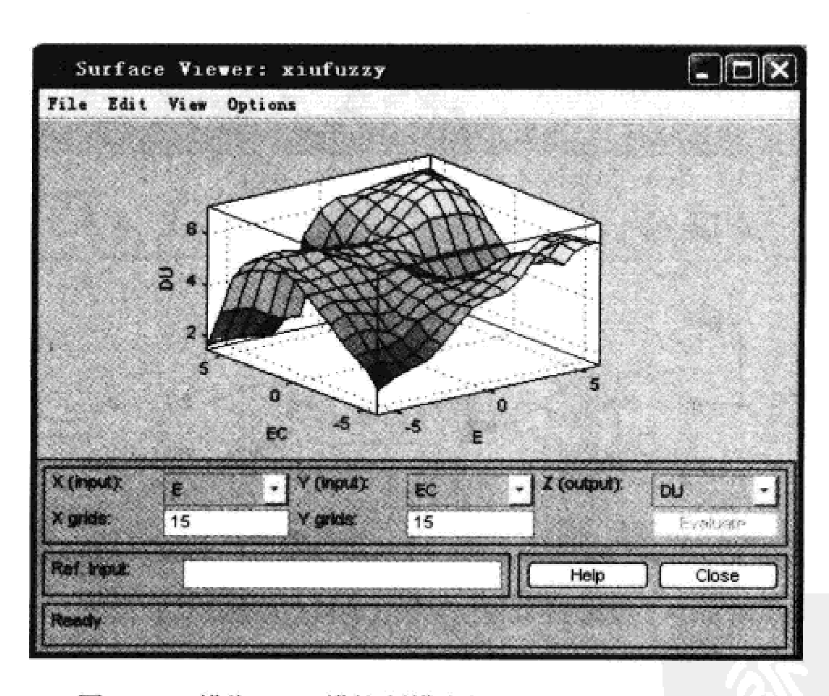


图 8-36 模糊-PI 双模控制模糊推理输入/输出关系曲面



3. 模糊-PI 双模控制 Simulink 建模仿真

打开 Simulink 工具箱，建立如图 8-37 所示的模糊-PI 双模控制系统仿真模型。图中模糊控制器的仿真模型如图 8-38 所示。在图 8-38 中双击“Fuzzy Logic Controller”模块，系统弹出模糊逻辑控制器参数设置对话框，如图 8-39 所示。在“FIS file or structure”文本框中输入已经建立的模糊推理系统文件名“xiufuzzy（注意 xiufuzzy 文件已载入 Workspace 中）”。因为，模糊逻辑工具箱与 Simulink 工具箱是通过工作空间变量传递数据的，因此在 Simulink 模型里使用模糊逻辑推理系统，调用的模糊推理系统文件名要与模糊推理系统编辑器输出到工作空间的文件名一致。设置仿真求解器为 ode45，仿真时间为 50s，最大仿真步

长设置为 0.01。其他参数取系统默认值，运行仿真，可得如图 8-40 所示的模糊-PI 双模控制系统仿真结果。

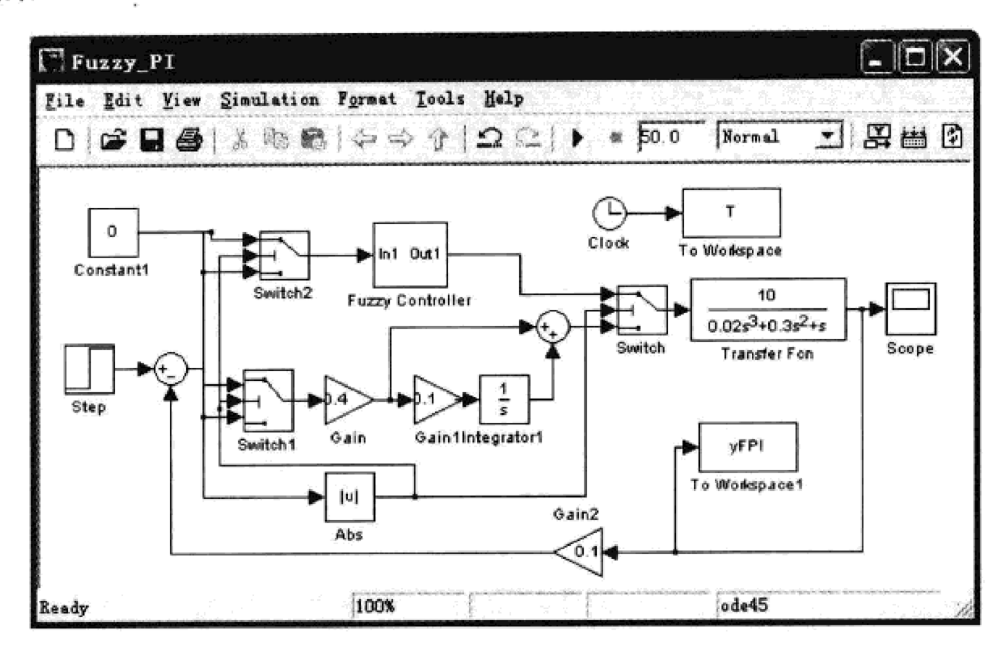


图 8-37　模糊-PI 双模控制 Simulink 仿真模型

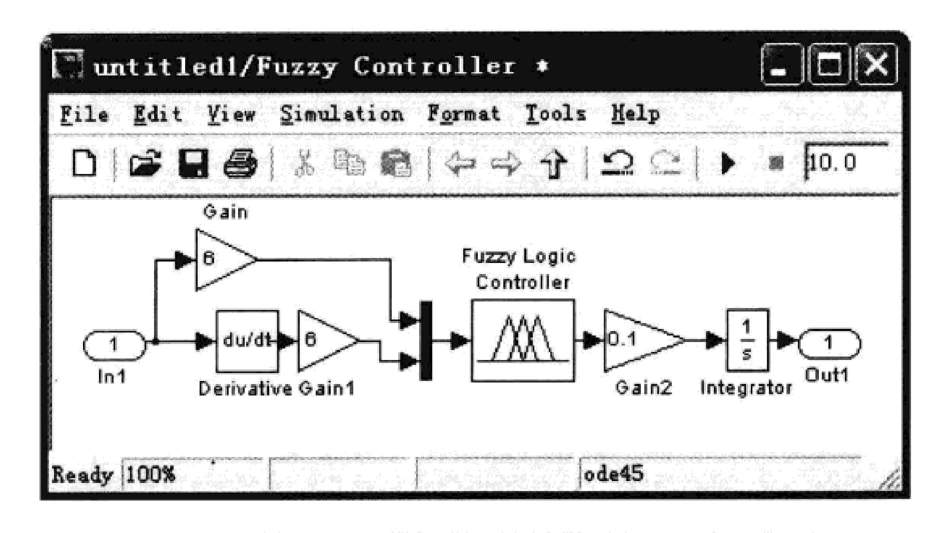


图 8-38　模糊-PI 双模控制系统的模型控制器仿真模型

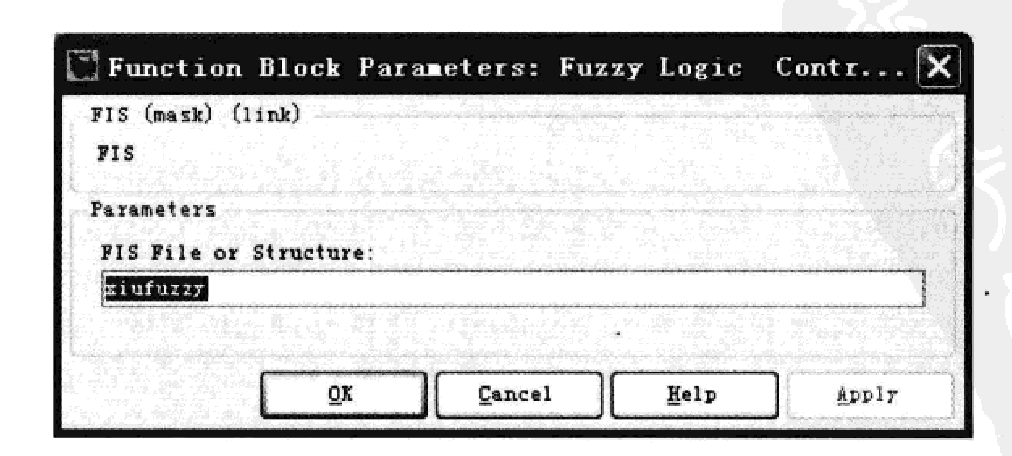


图 8-39　模糊控制器参数设置对话框

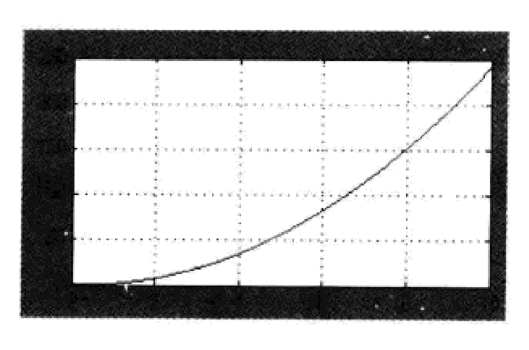


图 8-40 模糊-PI 双模控制系统仿真结果

8.5.3 模糊与 PID 双控制器仿真设计

将 PI 控制策略引入模糊控制器，构成模糊与 PI（或 PID）控制器复合控制，是改善模糊控制，是改善模糊控制器稳态性能的一种途径，控制器原理结构框图如图 8-41 所示。

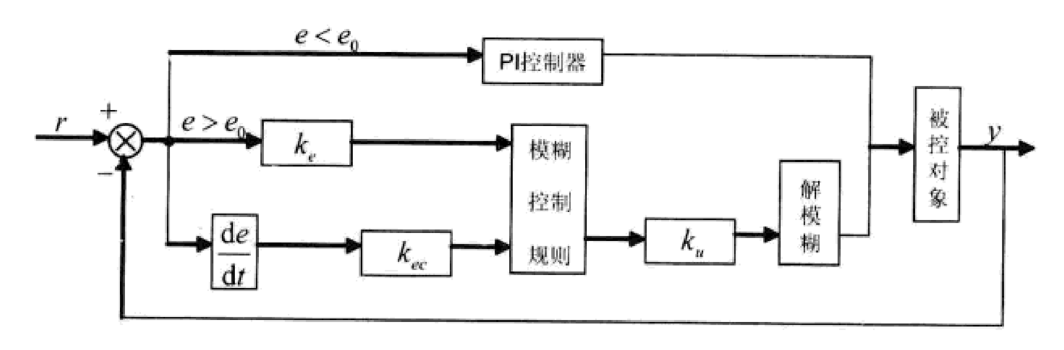


图 8-41 模糊与 PID 控制器原理结构

图中，r 为给定输入；y 为输出；e 为实际误差；e_0 为给定误差。针对其结构特点，在 Simulink 模型窗口中建立如图 8-42 所示模型。

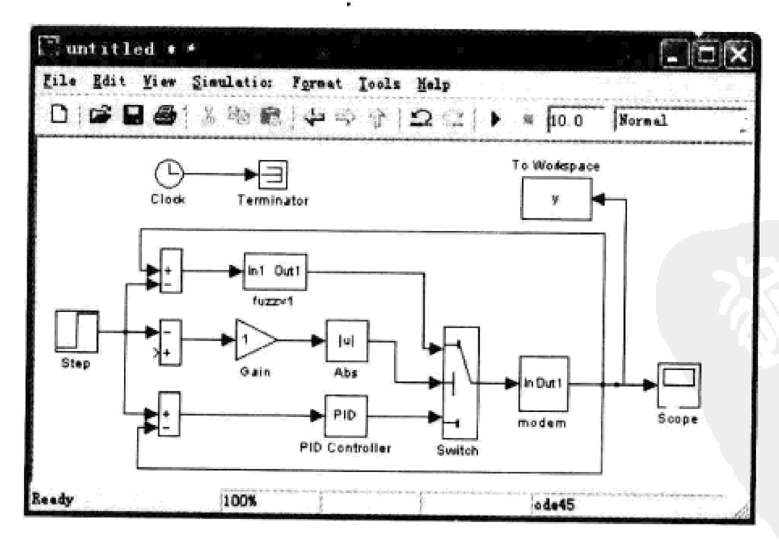


图 8-42 模糊与 PID 仿真模型

在图 8-42 中，fuzz1 和 modem 均为封装模块，其内部封装元件分别如图 8-43 及图 8-44 所示。

在命令窗口中输入 fuzzy，进入模糊推理系统编辑器，编辑输入、输出隶属度及模糊规则等项，完成后命名（如 xiufuzzy）存入相应的盘符中。双击图 8-43 中的“Fuzzy Logic

Controller”模块，将模糊推理编辑器的名字（xiufuzzy）写入其中，然后单击“OK”按钮，如图 8-45 所示。

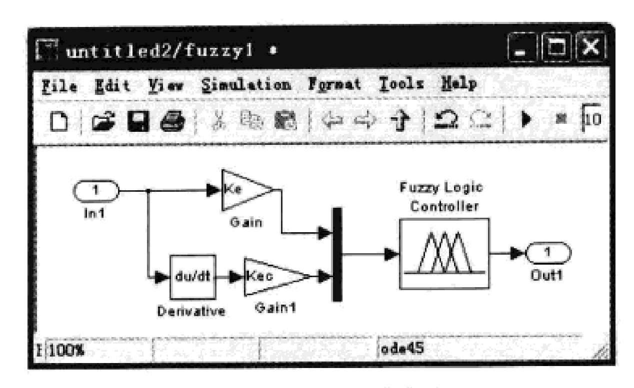


图 8-43 fuzzy1 内部结构

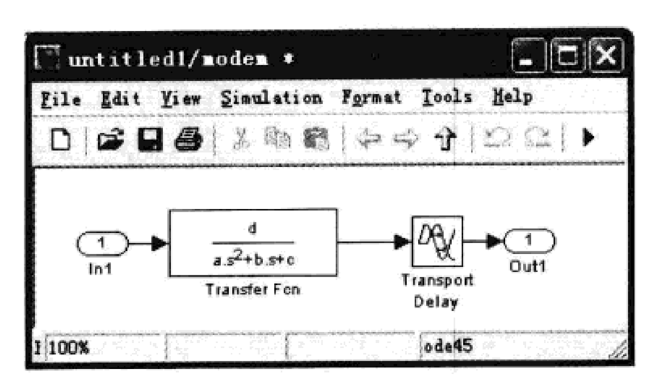


图 8-44 modem 内部结构

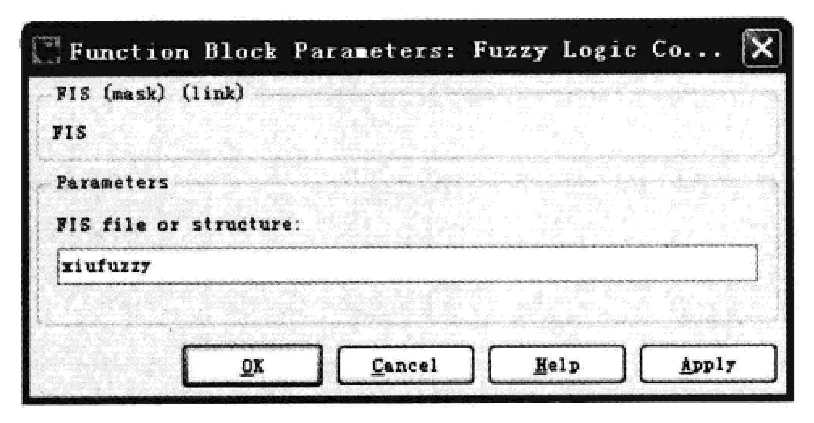


图 8-45 Fuzzy Logic Controller 命名

在图 8-45 中，k_e 和 k_{ec} 分别为误差、误差的变化的量化因子，k_u 为输出的比例因子，它们都根据实际情况的不同而取不同的值。信号源取单位阶跃响应信号（也可以取其他的信号源）。图 8-46 中，$\dfrac{d}{as^2+bs+c}$ 为实际的控制对象，a、b、c、d 为常数。Transport Delay 为滞后模块（可以根据实际需要添加）。仿真时间可以根据不同的控制对象来设定。

本例取 a、b、c、d 分别为 320、108、1 和 1，滞后时间取 25s，仿真时间取 50s，则得到的仿真结果如图 8-46 所示。

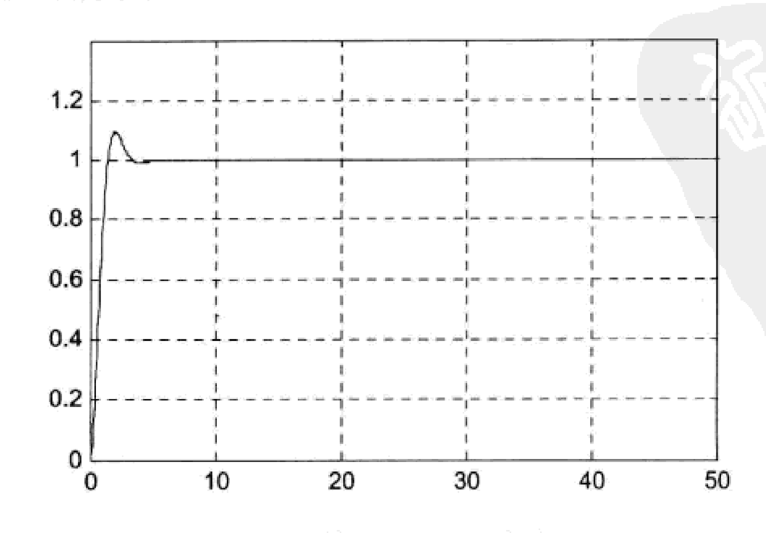


图 8-46 模糊与 PID 仿真结果

8.5.4 模糊-PID 控制器仿真设计

模糊-PID 控制器也称模糊自整定 PID 参数控制器。是在常规 PID 控制器的基础上，采用模糊逻辑推理方法来调整 PID 控制算法中的参数。经模糊推理得到的结果不是直接作为系统的输出，而是用该结果来整定 PID 的参数，再根据 PID 算法来决定系统的输出。其结构如图 8-47 所示。



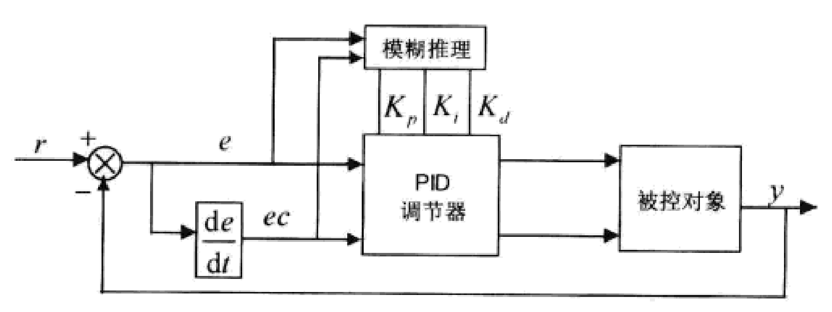


图 8-47　模糊-PID 控制器结构

图中，r、y 分别为给定输入、输出；e、ec 分别为误差和误差的变化；K_p 、K_i、K_d 分别为用于调速 PID 控制器的模糊控制输出，它们分别调整 PID 控制器的比例、积分和微分。

针对其结构特点，在 Simulink 模型窗口中建立如图 8-48 所示模型。

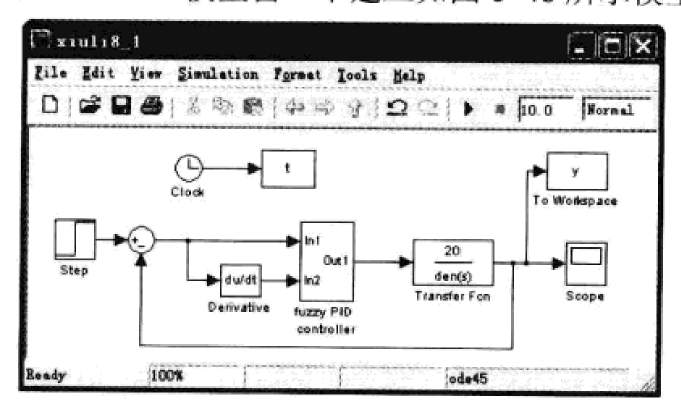


图 8-48　模糊 PID 控制器仿真模型

图中，$\dfrac{d}{as^2+bs+c}$ 为实际的控制对象，a、b、c、d 为常数。Fuzzy PID Controller 封装模块，其内部封装如图 8-49 所示，图中又包括两部分封装 Fuzzy Logic Controller 和 PID Controller，其内部分别如图 8-50 和图 8-51 所示，输入信号源取单位阶跃脉冲信号。

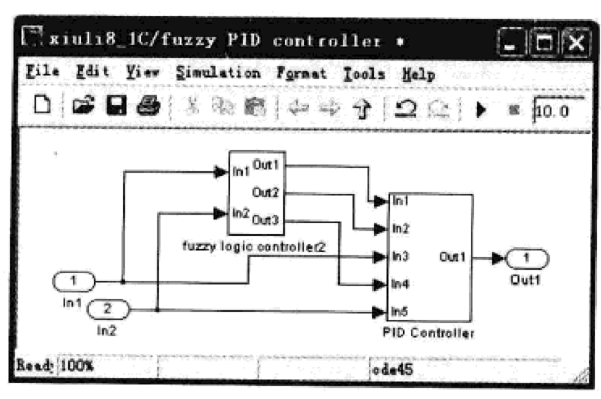


图 8-49　Fuzzy PID Controller2 的内部封装

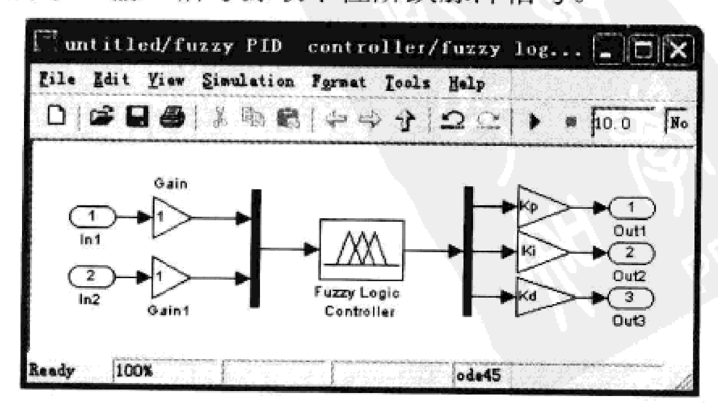


图 8-50　Fuzzy Logic Controller 的内部封装

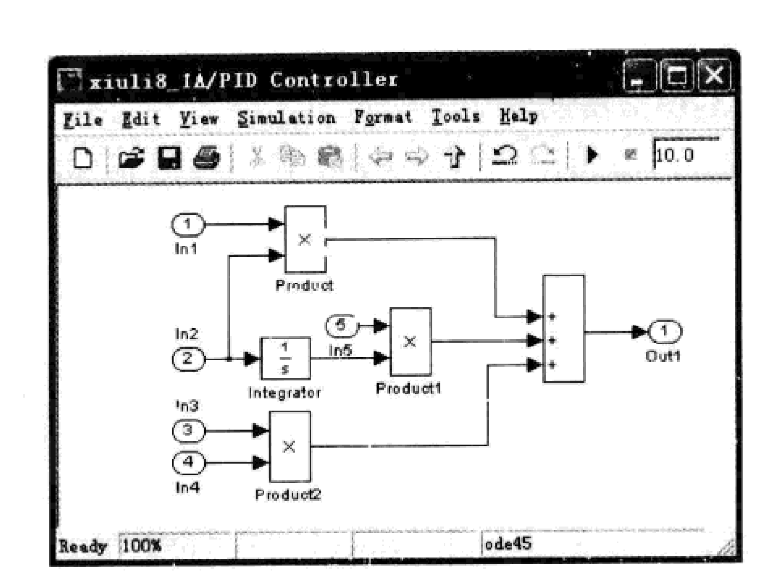


图 8-51 PID Controller 的内部封装

在命令窗口中输入 fuzzy，进入模糊推理系统编辑器，编辑输入、输出隶属度及模糊规则等项，完成后取名存入相应的盘符中。双击图 8-50 中的 Fuzzy Logic Controller，将模糊推理编辑器的名字写入其中，然后单击“OK”按钮。

本例取 a、b、c、d 分别为 1.6、4.4、1、20，仿真时间取 10s，则得到的仿真结果如图 8-52 所示。

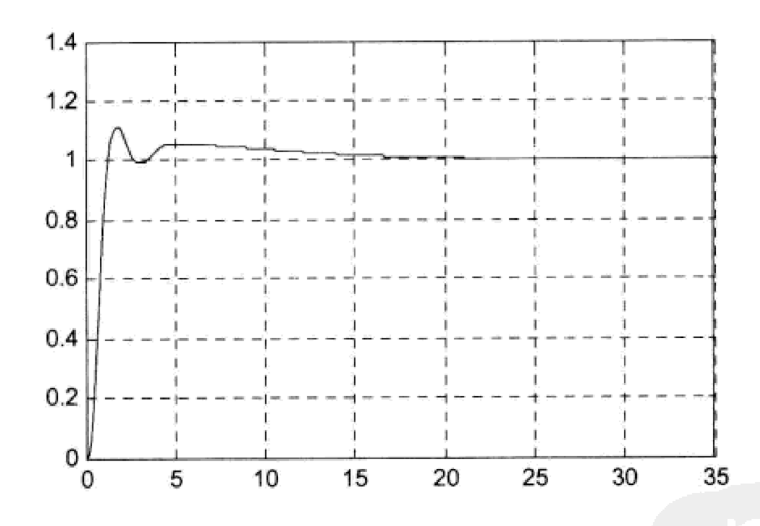


图 8-52 模糊-PID 仿真结果

第 9 章 Simulink 建模与仿真在通信系统中的应用



现在，通信技术与人们的生活联系越来越紧密，现代通信系统是信息时代的生命线，信息技术革命是伴随着通信技术的发展而进行的。从 19 世纪以来，通信技术发展很快，尤其是 20 世纪 50 年代后，在计算机的使用普及以后，数字通信在越来越多的领域取代了模拟通信，模拟调制技术也发展为脉冲编码调制等技术。

9.1 通信系统仿真方法介绍

9.1.1 求解动态系统建模的状态方程方法

动态系统，也就是有记忆系统的数学描述的状态方程。所谓系统建模，就是根据研究对象的物理模型找出相应的状态方程的过程。而所谓对动态系统的仿真，就是利用计算机来对所得出的状态方程进行数值求解的过程。在通信系统中，通常人们关心的信号是以时间为自变量的函数，所以相应状态方程中的状态变量、输入变量、输出变量也都是时间的函数 为叙述方便，文中假定状态方程都是基于时间的。

对于确定系统，当给定系统的初始状态和输入信号时，其输出信号也是确定的。在连续时间系统中，状态方程是一组微分方程。在当前时刻 t 处的状态向量值（注意高阶系统有可能是多个独立状态，故表示为向量）$s(t)$ 和输入信号向量值 $x(t)$ 已知的条件下，以微分方程组形式的状态方程确定了当前时刻输出信号向量 $y(t)$，以及“与当前时刻无限接近的下一时刻” $t+\mathrm{d}t$ 的新状态向量 $s(t+\mathrm{d}t)$。依此类推，如果已知当前系统的状态，由状态方程就能给出未来所有时刻上的系统状态值和输出信号值。

在计算机数值求解中，只能以一个微小的时间间隔 Δ 来近似表示当前时刻与下一时刻之间的无穷小时间差 $\mathrm{d}t$，所以数值求解（实质上就是微分方程的数值求解）总是近似的，这个微小的时间间隔 Δ 就称为求解的步长。在给定求解精度要求下，需要根据动态系统的性质以及输入信号的特征来选择求解步长和求解算法。通常，求解步长过小将增加计算量，使仿真速度下降，而求解步长太大会严重影响仿真结果的精度，甚至导致求解递推过程不收敛而使求解失败。

微分方程的求解算法可以划分为两大类：变步长算法和固定步长算法。在变步长算法中，求解步长是自适应变化的，以兼顾求解精度和求解速度。而固定步长算法中的步长需要在仿真之前根据系统特征、信号特征和精度要求进行设置。在通信系统中，流动的信号和相应的处理部分一般是频带受限的，由取样定理给出了保证连续信号离散化过程不失真所要求的最大取样间隔，所以只要固定步长算法的求解步长设定满足取样定理的要求，一般也就能够保证求解的正确性。

对于离散时间系统，状态方程以组差分方程的形式给出。求解就是要得出在各离散时刻$(0,1,2,\cdots,k,\cdots)$上的系统状态值和输出信号值。当给定当前离散时刻k处的状态向量值$s(k)$，以及当前输入的时间离散信号取值$x(k)$，由差分方程组就确定了当前系统输出信号取值$y(k)$，以及下一个时刻（$k+1$ 时刻）新的系统状态取值$s(k+1)$。如果已知系统的初始状态$s(0)$和输入的离散时间信号$x(k)$，$k=0,1,2,\cdots$，通过递推，就可以得到出未来各个离散时刻的系统状态值和系统输出信号。

如果系统模型中存在数模转换模块（如取样器、模拟低通滤波器等），那么系统中既存在时间连续信号，又有时间离散信号，其状态方程中既有微分方程，又有差分方程。对于这种混合系统，在进行数值求解时往往可根据取样定理，采样满足系统最高工作频率不失真要求的固定步长算法，这样易于协调微分方程和差分方程之间的数据交互。

【例 9-1】 对乒乓球的弹跳过程进行仿真。忽略空气对球的影响，乒乓球垂直下落，落点为光滑的水平面，乒乓球接触落点立即反弹。如果不考虑弹跳中的能量损耗，则反弹前后的瞬时速率不变，但方向相反。如果考虑撞击损耗，则反弹速率有所降低。目的是通过仿真得出乒乓球位移随时间变化的关系曲线，并进行弹跳过程的"实时"动画显示。

（1）数学模型

首先对乒乓球弹跳过程进行一些理想化假设。设球是刚性的，质量为m，垂直下落，碰击面为水平光滑平面。在理想情况下碰击无能量损耗。如果考虑碰击面损耗，则碰击前后速度方向相反，大小按比例系数 K，$0<K\leqslant 1$下降。在t时刻的速度设为$v=v(t)$，位移设为$y=y(t)$，并以碰击为坐标原点，水平方向为坐标横轴建立直角坐标系。球体的速度以竖直向上方向为正方向。重力加速度为$g=9.8\text{m/s}^2$。

初始条件假设：设初始时刻$t_0=0$，球体的初始速度为$v_0=v(t_0)$，初始位移为$y_0=y(t_0)$。

受力分析：在空中时小球受重力 $F=mg$ 作用，其中，$g=\dfrac{\mathrm{d}v}{\mathrm{d}t}$。则在$t+\mathrm{d}t$时刻小球的速度为（注意，其中负号是考虑了速度的方向）：

$$v(t+\mathrm{d}t)=v(t)-g\mathrm{d}t \tag{9-1}$$

在$t+\mathrm{d}t$时刻小球的位移为：

$$y(t+\mathrm{d}t)=y(t)+v(t)\mathrm{d}t \tag{9-2}$$

在小球撞击水平的瞬间，即$y(t)=0$的时刻，它的速度方向改变，大小按比例 K 衰减。当K=1 时，就是无损耗弹跳情况。因此，小球反弹瞬间（$t+\mathrm{d}t$ 时刻）的速度为：

$$v(t+\mathrm{d}t)=-Kv(t)-g\mathrm{d}t,\quad 0<K\leqslant 1 \tag{9-3}$$

反弹瞬间的位移为：

$$y(t+\mathrm{d}t)=y(t)-Kv(t)\mathrm{d}t=-Kv(t)\mathrm{d}t \tag{9-4}$$

（2）仿真模型设计

从数学模型中可知，小球在空中自由运动时刻与撞击时刻的动力方程不同。通过小球所处位置（位移）是否为零可判定小球处于何种状态。程序中采用 if 语句来作出判断，以决定使用式（9-1）还是式（9-3）来计算。其实现的 MATLAB 程序代码如下：



```
>> clear all;
g=9.8;                  %重力加速度
v0=0;                   %初始速度
y0=1.2;                 %初始位置
m=1.8;                  %小球质量
t0=0;                   %起始时间
K=0.85;                 %弹跳的损耗系数
n=5000;                 %仿真的总步长
dt=0.001;               %仿真步长
v=v0;                   %初状态
y=y0;
for k=1:n
    if(y>0)|(v>0)       %小球在空中的动力方程计算
        v=v-g*dt;
        y=y+v*dt;
    else                %如果碰击作如下计算
        y=y-K.*v*dt;
        v=-K.*v-g*dt;
    end
    s(k)=y;             %当前位移记录到 s 数组中以便作图
end
t=t0:dt:dt*(n-1);       %仿真时间
plot(t,s,'r:');
xlabel('时间/s');
ylabel('位移 y(t)/m');
axis([0 5 0 1.2]);
```

运行程序，效果如图 9-1 所示。图 9-1 中分别作出了碰击误差系数 K=1 和 K=0.85 两种情况下的小球弹跳位移曲线。对程序稍加修改就可以得到显示小球弹跳过程的动画。修改后的程序文件代码如下。有兴趣的读者可以修改程序，观察不同的碰击衰减系数下的小球弹跳过程。

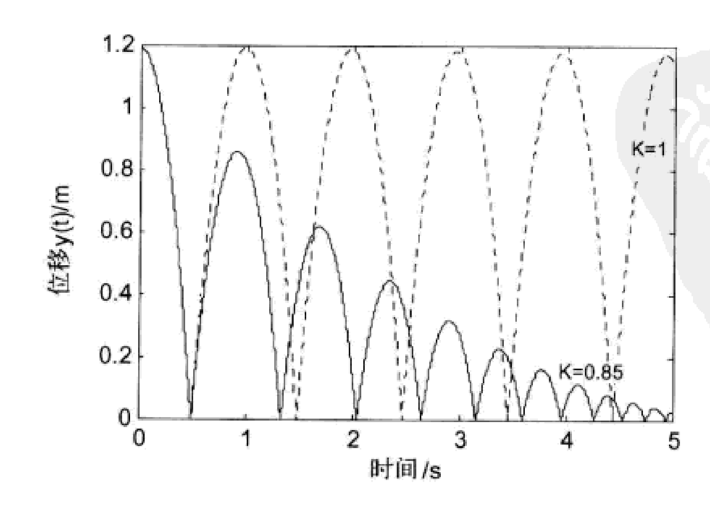


图 9-1　碰击衰减系数 K=0.85 和 K=1 情况下的小球弹跳位移效果

```
>> clear all;
g=9.8;                          %重力加速度
v0=0;                           %初始速度
y0=1.2;                         %初始位置
m=1.8;                          %小球质量
t0=0;                           %起始时间
K=0.85;                         %弹跳的损耗系数
n=5000;                         %仿真的总步长
dt=0.005;                       %仿真步长
v=v0;                           %初状态
y=y0;
for k=1:n
    if(y>0)                     %小球在空中的动力方程计算
        v=v-g*dt;
        y=y+v*dt;
    else                        %如果碰击作如下计算
        y=y-K.*v*dt;
        v=-K.*v-g*dt;
    end
    plot(t,s,'ro');
    axis([-2 2 0 1]);
    set(gcf,'DoubleBuffer','on');
    drawnow;
end
```

9.1.2 蒙特卡罗法

蒙特卡罗（Monte Carlo）方法是一种基于随机试验和统计计算的数值方法，也称计算机随机模拟方法或统计模拟方法。20 世纪 40 年代中期，在美国为第二次世界大战而研制原子弹的“曼哈顿计划”中，人们提出了以概率统计理论为指导的一类非常重要的数值计算方法，并用驰名世界的赌城——摩纳哥的蒙特卡罗来命名这种方法。现代 Monte Carlo 方法正是以概率为基础的方法。

Monte Carlo 方法的基本思想很早以前就被人们所认识和利用了。17 世纪，人们就知道用事件发生的“频率”来决定事件的“概率”。19 世纪人们用投针试验的方法来计算圆周率 π，这就是一种手工形式的 Monte Carlo 方法。随着计算机的出现，特别是近年来高速计算机的出现，使得在计算机上大量、快速地模拟这样的随机试验成为可能。

Monte Carlo 方法的数学基础是概率论中的大数定理和中心极限定理。大数定理指出，随着独立随机试验次数增加，试验统计事件出现的频率将接近于该统计事件的概率。Monte Carlo 方法的基本思想是，当所求解问题是某种随机事件出现的概率，或者是某个随机变量的期望值时，通过某种“实验”的方法，以这种事件出现的频率来估计该随机事件的概率，或者得出这个随机变量的某些数字特征，并将其作为问题的解。如果所求解的问题不是一个随机事件问题，那么可以通过数学分析方法找出与之等价的随机事件模型，然后再利用 Monte Carlo 方法去求解。

下面以一个直观的例子来解释 Monte Carlo 方法。假设要计算一个不规则图形的面积，那么图形的不规则程度和解析性计算（比如积分）的复杂程度是成正比的，如果图形足够复杂，将很难甚至不可能得出解析计算的表达式。Monte Carlo 方法是怎样计算的呢？设想有一袋豆子，把豆子均匀地朝这个图形上撒（假定豆子都在一个平面上，相互之间没有重叠），然后数这个图形之中有多少颗豆子，得出豆子数目就是图形面积。当豆子越小，撒得越多、越均匀时候，结果就越精确。以数学语言来描述就是：对于平面上一个边长为 1 的正方形及其内部一个形状不规则的“图形”，如何求出这个“图形”的面积呢？Monte Carlo 方法这么做：向该正方形均匀地随机投掷 M 个点，如果其中有 N 个点落于“图形”内，则该“图形”的面积近似为 N/M。投掷的点数越多，结果就越精确。

就通信系统而言，由于面临信道、噪声环境，以及系统本身的复杂性，使得求解问题的维数（即变量的个数）可能高达数百甚至上千。即使找到了解析结果，用传统的数值方法也难以计算（即使使用速度最快的计算机）。况且在更多情况下，当系统模型考虑了多种因素（即变量的个数）后，往往解析结果是难以得出的。对这类问题，求解的难度将随维数的增加呈指数增长，即导致所谓的“维数灾难”。 Monte Carlo 方法的计算复杂性不再依赖于维数，也不需要知道问题的解析表达式，因此能够很好地用来对付“维数灾难”问题和解析结果未知的问题。利用高速计算机，以前这些本来无法计算的问题现在也能够得出数值结果了。

在建模和仿真中，应用 Monte Carlo 方法主要有两部分工作。

- 用 Monte Carlo 方法模拟某一过程，产生所需要的各种概率分布的随机变量。
- 用统计方法把模型的数字特征估计出来，从而得到问题的数值解，即仿真结果。

下面给出一个用来计算圆面积的 Monte Carlo 方法仿真示例。

【例 9-2】 试用 Monte Carlo 方法求出半径为 1 的圆的面积，并与理论值对比。

（1）数学模型

设有两个相互独立的随机变量 x, y，服从[0 2]上的均匀分布。那么，由它们所确定的坐标点 (x, y) 是均匀分布于边长为 2 的一个正方形区域中，该正方形的内接圆的半径为 1，如图 9-2 所示。显然，坐标点 (x, y) 落入圆中的概率 p 等于该圆面积 S_c 与正方形面积 S 之比，即

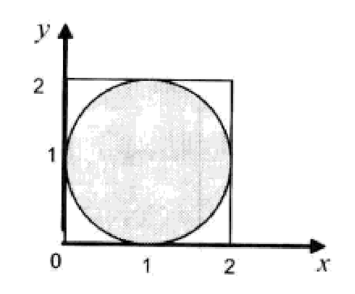


图 9-2 用 Monte Carlo 方法求圆面积

$$S_c = pS \tag{9-5}$$

因此，只要通过随机试验统计算出落入圆点的频度，即可计算出圆的近似面积来。当随机试验的次数充分大的时候，计算结果就趋近于理论真值。

（2）仿真试验

其实现的 MATLAB 程序代码如下：

```
>> clear all;
s=0:0.01:2*pi;
x=sin(s);
y=cos(s);                    %计算半径为 1 的圆周上的点,以便作出圆周观察
m=0;                         %在圆内在落点计数器
x1=2*rand(999,1)-1;          %产生均匀分布于[-1 1]直接的两个独立随机数 x1,y1
```

```
y1=2*rand(999,1)-1;
N=999;                                   %设置试验次数
for n=1:N                                %循环进行重复试验并统计
    p1=x1(1:n);
    q1=y1(1:n);
    if(x1(n)*x1(n)+y1(n)*y1(n))<1        %计算落点到坐标原点的距离,差别落点是否在圆内
    m=m+1;                               %如果落入圆中,计数器加 1
    end
    plot(p1,q1,'.',x,y,'-k',[-1 -1 1 1 -1],[-1 1 1 -1 -1],'-k');
    axis equal;                          %坐标纵横比例相同
    axis([-2 2 -2 2]);                   %固定坐标范围
    text(-1,-1.2,['试验总次数 n=',num2str(n)]);    %显示试验结果
    text(-1,-1.4,['落入圆中数 m=',num2str(m)]);
    text(-1,-1.6,['近似圆面积 Sc=',num2str(m/n*4)]);
    set(gcf,'DoubleBuffer','on');
    drawnow;
end
```

程序执行中，将动态显示随机落点情况和当前的统计计算结果。图 9-3 为重复落点 288 次时的计算结果。随着试验次数增加，计算结果将趋近于半径为 1 的圆面积的真值 π 。

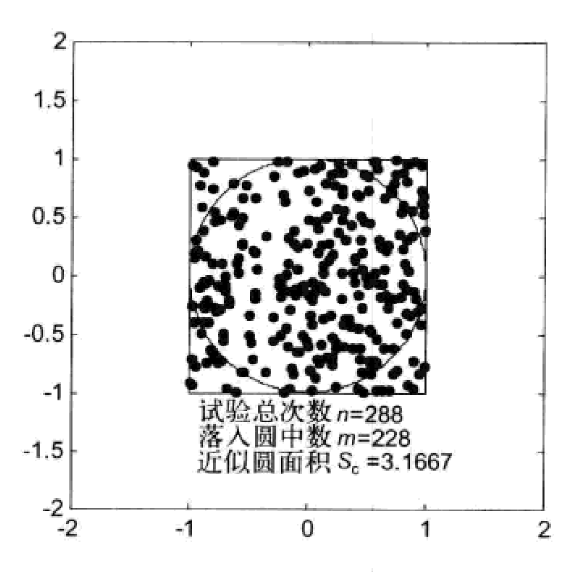


图 9-3 Monte Carlo 方法随机试验计算圆面积的过程

动画模式适合于原理演示。但是，如果要提高程序效率，就应该取消仿真过程中的可视化显示，并利用 MATLAB 的矩阵运算机制来改造程序。下面的程序将随机试验次数提高到了 1000 万次，计算得到的圆面积（也即圆周率）精度提高到了小数点后 2 位。程序中同时使用了矩阵运算机制和循环结构来负责完成重复随机试验，其目的是为了兼顾计算速度和程序内存占用量。矩阵运算是一种并行计算机制，计算速度快，但是矩阵越大，内存占用就越多；而循环结构则可重复使用相同的内存区域，尽管速度较慢。这是 MATLAB 语言固有的特点，在编程中应当就具体问题作出权衡。

```
>> tic                                   %启动计时器
n=10000;                                 %每次随机落点 10000 个
for k=1:1000                             %重复试验 1000 次
    x1=2*rand(n,1)-1;
    y1=2*rand(n,1)-1;
    m(k)=sum((x1.*x1+y1.*y1)<1);         %求落入圆中的点数和
end
Sc=mean(m).*4./n                         %计算并显示结果
time=toc                                 %显示耗时
```

由于是随机试验，重复运行的结果也不完全相同，且不同计算机配置上的运行耗时也不一样，运行结果如下：

```
Sc =
     3.1422
time =
     238.8182
```



9.1.3 混合法

在实践中，我们往往首先根据研究目的、系统结构，以及所需要得出的系统参数等指标来建立相应的仿真模型。如果系统属于动态系统，在数学上即用状态方程描述，那么对该系统的仿真过程就是求解该微分方程组的过程。然而，许多时候人们希望考察系统在具有随机性的环境中表现。例如，研究系统的老化过程、热稳定性，以及系统对噪声的处理情况等，这时系统模型的参数（如输入信号、方程系数等）将含有随机成分，那么对系统的仿真就是在具有随机变量条件下的微分方程数值求解问题，这样的仿真方法就称为混合方法。因为仿真同时使用了基于数值计算的状态方程求解方法和基于统计计算的 Monte Carlo 方法。由于通信系统是一种工作在随机噪声环境下的动态系统，所以对通信系统的一般仿真方法就是确定方程求解与统计计算相互结合的混合方法。

这里需要指出，并非任何计算数值求解过程都可以看做系统的仿真过程。如果计算是对理论所得出的解析公式的数值计算，那么这种计算就不是仿真。例如，欲求解某动态系统的阶跃响应，可以先建立该系统的状态方程，然后通过数学方法（例如，若是线性时不变系统，可用拉普拉斯变换方法求解）求出系统的阶跃响应的解析表达公式，再通过计算机编程计算得出解析公式的数值结果，并画出曲线，但是这仅仅是对理论解的数值计算而已。如果在建立了系统的状态方程之后，定义输入信号为阶跃函数，然后直接对状态方程作数值计算得出结果，那么这就是一个仿真过程。又比如，在加性高斯信道条件下，数字通信系统的传输误码率与信噪比之间的关系可以通过概率分析的方法得到解析公式，根据误码率解析公式计算得出结果（曲线）的过程仅仅是解析数值计算过程，不是系统仿真的过程。而通过 Monte Carlo 方法对传输进行试验并进行误码统计得出结果（曲线）的过程就是仿真过程。

显然，如果解析数值计算和仿真过程都是正确的，那么在误差范围内，两者所得出的结果必然是一致的，这样就可以通过仿真结果与解析结果之间的对比来检验程序的正确性。可见，对系统的仿真只需要建立系统的数学模型，而不需要对模型的理论求解（在实际问题中，往往理论求解是不可能的或不存在的，例如将上述系统的输入信号变为随机噪声，或者将上述系统变为一个时变系统或非线性系统）。因此，当验证了仿真计算过程的正确性之后，可以将之推广到更为复杂或更加接近实际的情况，从而得出通过解析方法难以得到的数值结果。下面将例 9-1 推广到更加接近真实物理模型的情况。

【例 9-3】 实际物理试验中，当一个乒乓球垂直下落到一个完全水平的玻璃板上后，乒乓球不断弹跳，直到能量耗尽。假定空气是静止的，没有风，但弹跳中的乒乓球在玻璃板上的落点仍不会是同一点，这说明在乒乓球运动过程中受到微弱的水平面方向力的作用，产生了水平方向上的漂移。这些水平力在示例 9-1 中被忽略不计，所以那里仿真的结果中小球落

点总是在坐标原点处。如果要建立更加接近真实物理环境的弹跳模型，就必须考虑这些被忽视的微小的扰动因素。通过物理实验观察，我们可以做这样的合理假设：水平面方向上对乒乓球的微弱作用力可能来自多种因素的综合，其中各因素对合力的贡献甚小。根据大数定理，在数学上就可以将水平作用力建模为一个高斯随机变量。为简单起见，这里仍然忽略了空气对小球的其他作用因素，如球运动中的阻力、空气的浮力等。

同时，将例 9-1 推广到三维空间中的情况。

设水平面为 xz 坐标平面，y 轴指向为垂直方向。小球在 x 方向上的受力 $F_x(t)$ 是一个零均值独立高斯随机过程。小球在 z 方向上的受力 $F_z(t)$ 与 $F_x(t)$ 具有相同的分布，但两者相互独立。即

$$F_x(t) \sim N(0,\sigma^2) \tag{9-6}$$

$$F_z(t) \sim N(0,\sigma^2) \tag{9-7}$$

x 、z 方向相应的加速度、速度和位移分别用 a_x 、a_z 、v_x 、v_z 、s_x 、s_z 表示，小球的质量为 m 。由牛顿第二运动定律，可得出以下运动方程。

$$\begin{aligned} a_x(t) &= F_x(t)/m \\ \mathrm{d}v_x(t) &= a_x(t)\mathrm{d}t \\ \mathrm{d}s_x(t) &= v_x(t)\mathrm{d}t \end{aligned} \tag{9-8}$$

z 方向的运动方程类似。据此编写 MATLAB 的仿真程序代码如下：

```
>> clear all;
g=9.8;                          %重力加速度
v0=0;                           %初始速度
y0=1.2;                         %初始位置
m=0.4;                          %小球质量
t0=0;                           %起始时间
K=0.85;                         %弹跳的损耗系数
n=5000;                         %仿真的总步长
dt=0.005;                       %仿真步长
v=v0;                           %初状态
y=y0;
vx=0;
vz=0;
sx=0;
sz=0;
for k=1:n
    if y>0                      %小球在空中的动力方程计算
        v=v-g*dt;
        y=y+v*dt;
    else                        %如果碰击作如下计算
        y=y-K.*v*dt;
        v=-K.*v-g*dt;
    end
   Fx=randn;                    %x 水平方向的随机力,方差为 1
   ax=Fx./m;                    %Fx 导致的 x 水平方向的加速度
```

```
    vx=vx+ax*dt;                    %小球在 x 水平方向的瞬时速度
    sx=sx+vx*dt;                    %小球在 x 水平方向上的位移
    Fz=randn;                       %z 水平方向的随机力,方差为 1
    az=Fz./m;                       %Fz 导致的 z 水平方向的加速度
    vz=vz+az*dt;                    %小球在 z 水平方向的瞬时速度
    sz=sz+vz*dt;                    %小球在 z 水平方向上的位移
    plot3(sx,sz,y,'r.');
    grid on;hold on;
    axis([-2 2 -2 2 0 1]);          %坐标范围固定
    set(gcf,'DoubleBuffer','on');   %双缓冲避免作图闪烁
    xlabel('水平方向 x');ylabel('水平方向 z');
    zlabel('垂直方向 y');title('小球的弹跳过程');
    drawnow;
end
```



仿真以动画方式进行，以便于观察。图 9-4 是程序运行的结果。图中显示了小球的运动轨迹，弹跳的落点是随机的。修改小球的质量，弹跳落点的概率特性也会发生变化，质量大的球落点相对集中。读者也可将空气阻力考虑到数学模型中，从而仿真出比较真实的弹跳过程。

从该例子中可以看出，计算机仿真方法可以在不知道解析求解的情况下通过“计算机试验”来研究事物的变化规律，方便人们研究更真实、更复杂的物理系统。往往这些考虑了多种因素的物理系统是很难进行解析分析的，这时仿真方法几乎就成为唯一能够获得求解的方法。在这个例子中，既用到了确定系统的微分方程求解，也用到了随机统计试验，这就是一种混合的仿真方法

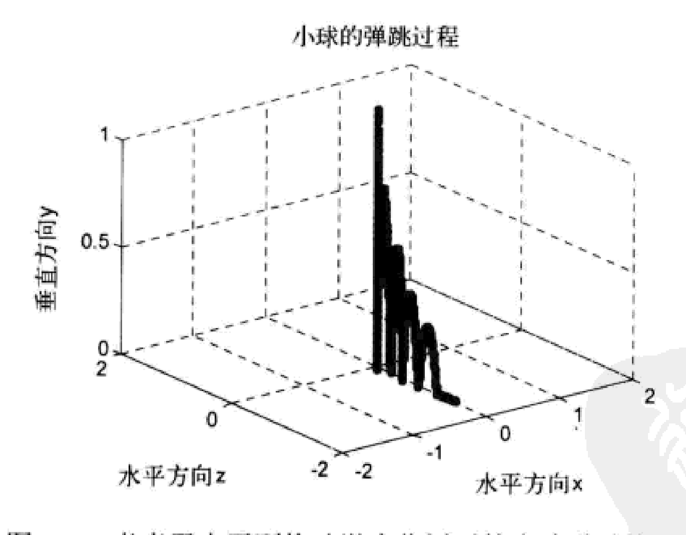


图 9-4　考虑了水平面扰动微力作用后的小球弹跳轨迹

9.2　信源与信道模型

通信系统一般由 3 部分组成，即信源、信道和信宿。信源是通信系统的起点，它产生数据并且对这些数据进行编码和调制，产生适合于信道传输的调制信号；信道是数据信号的传输载体，发送端发生的数据通过信源编码和信号调制转化成调制信号，然后进入信道。这些

调制信号通过信道到达接收端，在接收端通过与发送端相反的过程得到原始数据。信宿则是通信系统的终点，它从信道中接收信号，通过解码和解调得到信源端产生的原始数据。

信源、信道和信宿是通信系统中不可缺少的 3 部分。

9.2.1 随机数产生器

1. 随机数的功能与原理

随机整数产生器是用来产生[0，M-1]范围内具有均匀分布的随机整数。

随机整数产生器输出整数的范围[0，M-1]可以由用户自己定义。M 的大小可随机整数产生器中的“M-ary number”选项中随机输入。M 可以是标量也可以是矢量。如果 M 为标量，那么输出均匀分布且互不相关的随机变量。如果 M 为矢量，其长度必须和随机整数产生器中“Initial seed”的长度相同，在这种情况下，每一个输出对应一个独立的输出范围。如果“Initial seed”是一个常数，那么产生的噪声是周期重复的。

随机整数产生器的输出信号，可以是基于帧的矩阵、基于采样的行向量或列向量，也可以是基于采样的一维序列。输出信号的性质可以由“Frame-based outputs”、“Samples per frame”和“Interpret vector parameters as 1-D”3 个选项控制。

2. 随机数产生器的参数说明

随机整数产生器模块及参数设置对话框如图 9-5 所示。

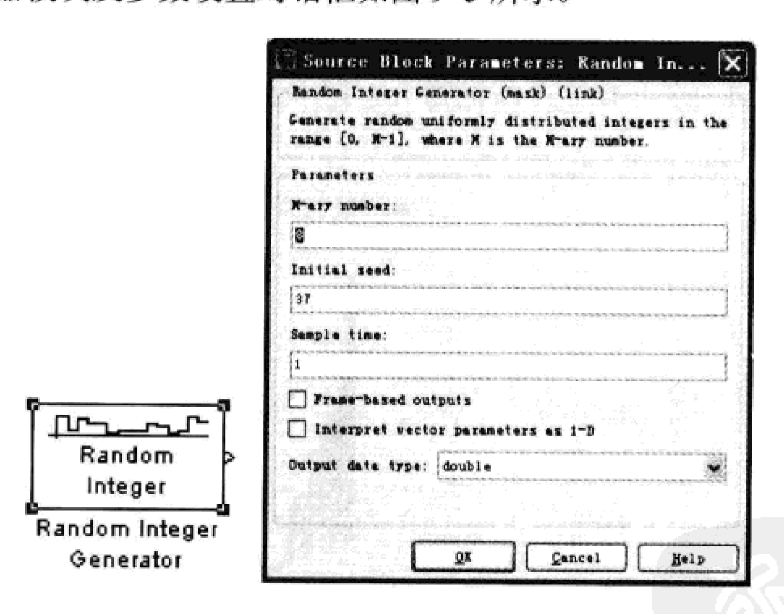


图 9-5　随机整数产生器模块及其参数设置框

随机整数产生器对话框包含多个参数项，下面分别对各项进行简单的介绍。

- “M-ary number”文本框：输入正整数或正整数矢量，设定随机整数的取值范围。当该参数设置为 M 时，随机整数的取值范围是[0，M-1]。
- “Initial seed”文本框：随机整数产生器的随机种子。当使用相同的随机数种子时，随机整数产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不能的序列。当随机数种子的维数大于 1 时，随机整数产生器的输出信号的维数也大于 1。
- “Sample time”文本框：输出序列中每个整数的持续时间。
- “Frame-based output”复选框：指定随机整数产生器以帧格式产生输出序列。即决定

输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”复选框未被选中时有效。

- “Sample per frame”：该参数用来确定每帧的抽样点的数目。本项只有当“Frame-based outputs”复选框被选中后有效。
- “Interpret vector parameter as 1-D”复选框：如果选中此复选框，则泊松分布整数产生器输出一维序列，否则输出二维序列。本项只有当“Frame-based output”复选框未被选中时有效。
- “Output data type”下拉列表框：决定模块输出的数据类型，可以是 boolean、int8、uint8、int16、uint16、int32、uint32、single、double 等众多类型，默认为“double”。如果想要输出为 boolean 型，“M-ary number”文本框必须为 2。



9.2.2 泊松分布产生器

1. 泊松分布产生器的功能与原理

泊松分布整数产生器产生服从泊松分布的整数序列。

泊松分布整数产生器利用泊松分布产生随机整数。假设 x 是一个服从泊松分布的随机变量，那么 x 等于非负整数 k 的概率可以用式（9-8）表示。

$$P_r(k)=\frac{\lambda^k \mathrm{e}^{-k}}{k!},\quad k=0,1,2,\cdots \tag{9-9}$$

式中，λ 为一正数，称为泊松参数。并且泊松随机过程的均值和方差都等于 λ。

利用泊松分布整数产生器可以在双传输通道中产生噪声，在这种情况下，泊松参数 λ 应比 1 小，通常远小于 1。泊松分布参数产生器的输出信号，可以是基于帧的矩阵、基于采样的行向量或列向量，也可以是基于采样的一维序列。输出信号的性质可以由泊松分布整数产生器中的“Frame-based outputs”、“Samples per frame”和“Interpret vector parameters as 1-D”3 个选项控制。

2. 泊松分布参数项说明

泊松分布整数产生器模块及其参数设置对话框如图 9-6 所示。

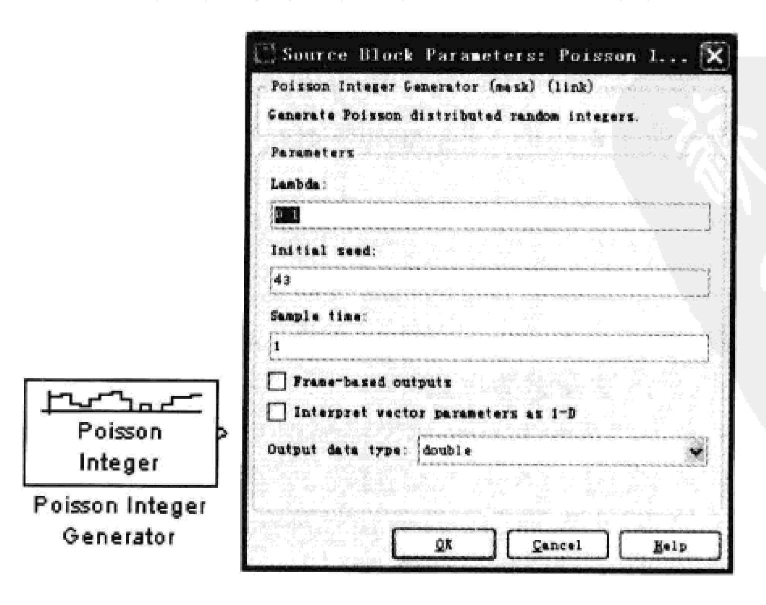


图 9-6 泊松分布整数产生模块及其参数设置对话框

泊松分布整数产生器对话框中包含多个参数项，下面分别对各项进行简单的说明。

- “Lambda”文本框：确定泊松参数λ，如果输入为一个标量，那么输出矢量的每一个元素分享相同的泊松参数。
- “Initial seed”文本框：泊松分布整数产生器的随机数种子。当使用相同的随机数种子时，泊松分布整数产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，泊松分布参数产生器的输出信号的维数也大于 1。
- “Sample time”文本框：输出序列中每个整数的持续时间。
- “Frame-based outputs”复选框：指定泊松分布整数产生器以帧格式输出序列，即决定输出信号是基于帧还是基于采样。本项只有当“Interpret vector parameters as 1-D”复选框未被选中时有效。
- “Samples per frame”复选框：该参数用来确定每帧的抽样点的数目。本项只有当“Frame-based outputs”复选框选中后才有效。
- “Interpret vector parameters as 1-D”复选框：如果选中此复选框，则泊松分布整数产生器输出一维序列。否则，输出二维序列。本项只有当“Frame-based outputs”复选框未被选中时有效。
- “Output data type”下拉列表框：决定模块输出的数据类型，可以是 boolean、int8、uint8、int16、uint16、int32、uint32、single、double 等众多类型，默认为“double”。

9.2.3 伯努利产生器

1. 伯努利二进制产生器功能与原理

伯努利二进制信号的产生器符合伯努利分布的随机信号。

伯努利二进制信号产生器产生随机二进制序列，并且在这个二进制序列中的 0 和 1 满足伯努利分布，如式（9-9）所示。

$$P_r(x)=\begin{cases} p, & x=0 \\ 1-p, & x=1 \end{cases} \tag{9-10}$$

即伯努利二进制信号产生器产生的序列中，产生 0 的概率为p，产生 1 的概率为$1-p$。根据伯努利序列的性质可知，输出信号的均值均为伯努利$1-p$，方差为$p(1-p)$。产生 0 的概率p由伯努利二进制信号产生器中的“Probability of a zero”控制，它可以是 0 和 1 之间的某个实数。

伯努利二进制信号产生器的输出信号，可以是基于帧的矩阵、基于采样的行向量或列向量，或者基于采样的一维序列。输出信号的性质可以由二进制伯努利序列产生器中的“Frame-based outputs”、“Samples per frame”和“Interpret vector parameters as 1-D”3 个选项控制。

2. 伯努利二进制信号产生器参数项说明

伯努利二进制信号产生器模块及其参数设定如图 9-7 所示。

伯努利二进制信号产生器中包含多个参数项，下面分别对各项进行简单的介绍。

- “Probability of a zero”文本框：伯努利二进制信号产生器输出“0”的几率。对应于式（9-9）中的p，为 0 和 1 之间的实数。
- “Initial seed”文本框：伯努利二进制信号产生器的随机数种子，它可以是与

"Probability of a zero"长度相同的矢量或标量。当使用相同的随机数种子时，伯努利二进制信号产生器每次都会产生相同的二进制序列；不同的随机数种子通常产生不同的序列。当随机数种子的维数大于 1 时，伯努利二进制信号产生器的输出信号的维数也大于 1。



- "Sample time"文本框：输出序列中每个二进制符号的持续时间。
- "Frame-based outputs"复选框：指定伯努利二进制信号产生器以帧格式产生输出序列。即决定输出信号是基于帧还是基于采样。本项只有当"Interpret vector parameters as 1-D"复选框未被选中时有效。
- "Interpret vector parameters as 1-D"复选框：如果选中此复选框，则伯努利二进制信号产生器输出一维序列。否则，输出二维序列。本项只有当"Frame-based outputs"复选框未被选中时有效。
- "Output data type"下拉列表框：决定模块输出的数据类型，可以是 boolean、int8、uint8、int16、uint16、int32、uint32、single、double 等众多类型，默认为"double"。

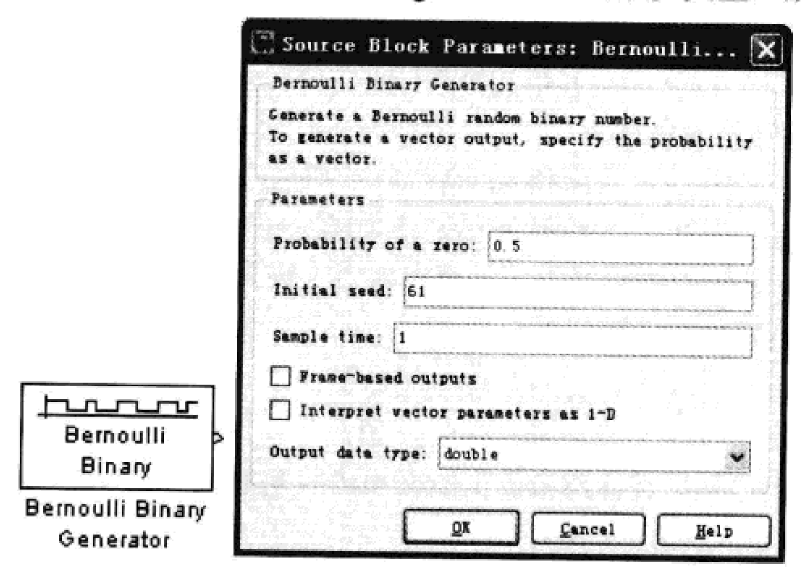


图 9-7 伯努利二进制信号产生模块及其参数设置对话框

9.2.4 加性高斯白噪声信道

如果噪声的取值服从零均值高斯分布，而任意不同时刻的取值相互独立，则称这样的噪声信号为高斯白噪声（AWGN）。高斯白噪声的自相关函数为一个冲激函数，其功率谱密度函数为常数。

在加性高斯白噪声信道中，信道的输入信号 $s(t)$ 将与信号内的零均值高斯白噪声 $n(t)$ 相叠加，得出输出信号 $r(t)$，即

$$r(t) = s(t) + n(t) \tag{9-11}$$

式中，输入信号 $s(t)$ 可以是实信号，也可以是复信号。当输入信号为实信号时，相叠加的高斯白噪声也是实信号，因此信道输出信号也是实信号，所叠加的零均值高斯白噪声的双边功率谱密度为 $N_0/2$（W/Hz）。如果输入信号为复信号，则所叠加的零均值高斯白噪声也是复信号，其实部和虚部相互独立且各自的功率谱密度相等，为 $N_0/2$（W/Hz）。因此，复高斯白噪声的功率谱密度为 N_0（W/Hz）。

9.2.5 错误概率信道

以上的信道模型是以信号波形和噪声波形的关系来建立的，这样的信道模型称为波形信道模型。对于数字通信系统，不同传输的波形代表了不同的传输信息符号，接收端收到被噪声污染的传输波形后，对其进行判决得出所代表的传输信息符号。由于噪声的影响，接收端判决输出的信息符号可能会与发送的信息符号不同，即可能以某种概率出现判决错误。可以将发送信息符号的输出端口视为信道的输入点，而将接收判决输出作为信道的输出点，那么信道输入/输出关系可以用输入/输出符号之间的错误概率关系来建立。直接以信道输入符号和输出符号之间的错误概率关系建立的信道模型称为错误概率信道模型。

当信道输出的信息符号之间相互没有影响，即相互独立时，则称这样的信道为离散无记忆信道（DMC）。离散无记忆信道可以用信道转移概率矩阵来表示。设信道输入符号集合为 $\chi=\{x_1,x_2,\cdots,x_j,\cdots,x_N\}$ ，并设信道输出的符号集合为 $\gamma=\{y_1,y_2,\cdots,y_i,\cdots,y_M\}$ ，在发送符号 x_j 的条件下，相应接收符号为 y_i 的概率记为 $P(y_i\,|\,x_j)$ ，称为信道转移概率。由信道转移概率构成信道转移概率矩阵，记为：

$$P=[P(y_i\,|\,x_j)]=\begin{pmatrix} P(y_1\,|\,x_1) & \cdots & P(y_1\,|\,x_N) \\ \vdots & \ddots & \vdots \\ P(y_M\,|\,x_1) & \cdots & P(y_M\,|\,x_N) \end{pmatrix} \tag{9-12}$$

二进制对称信道（BSC）是离散无记忆信道的一个特例，其输入/输出符号集合分别为 $\chi=\{0,1\},\gamma=\{0,1\}$ ，传输中由 0 错为 1 的概率与由 1 错为 0 的概率相等，设为 p 。那么，二进制对称信道的信道转换概率矩阵为：

$$P=\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \tag{9-13}$$

人们也经常用信道概率转换图来等价地表示离散无记忆信道，如二进制对称信道，如图 9-8 所示。

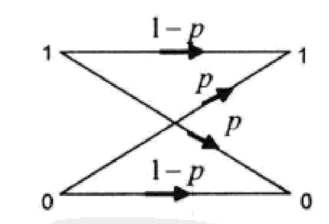


图 9-8　二进制对称信道模型

Simulink 通信模块库中提供了二进制对称信道模块 Binary Symmetric Channel，可设置错误概率，以及随机化种子，该模块还可以选择显示误码序列的输出端口。

下面的示例演示了二时制对称信道的建模和错误概率统计模块的使用情况。

【例 9-4】利用 Binary Symmetric Channel 模块和 Simulink 基本模块构建等价的二进制对称信道，设传输错误概率为 0.013，用 Error Rate Calculation 统计误码率。传输信号为二进制单极性信号，用 Bernoulli Binary Generator 模块产生，传输位率为 1000bit/s。

根据题设建立的模型如图 9-9 所示。

由于传输位率为 1000bit/s，所以采用步长为 0.001s 的固定步长仿真算法，并将 Bernoulli Binary Generator 模块的采样率也设置为 0.001s。二进制对称信道模型采用两种等价方式实现：一种是直接利用通信模块库中的 Binary Symmetric Channel 实现，根据题设要求将其误码参数设置为 0.013；另外一种方法是利用 Simulink 基本模块构建：通过设置在 0～1 之间均匀随机数发生模块、常数模块，以及一个关系模块 Relational Operator 得出以常数模块指定的常数作为概率的误码序列，如果随机数小于指定常数，则“<”关系成立，Relational

Operator 输出为 1，否则输出为 0，这样误码输出序列中 1 的概率就等于常数模块的设置值。然后，通过异或逻辑（XOR）模块将误码加入到传输信号中，得到布尔数据类型的输出序列。由于误码率计算模块 Error Rate Calculation 的输入数据类型要求是双精度类型的，所以还要用 Data Type Conversion 进行数据类型转换。



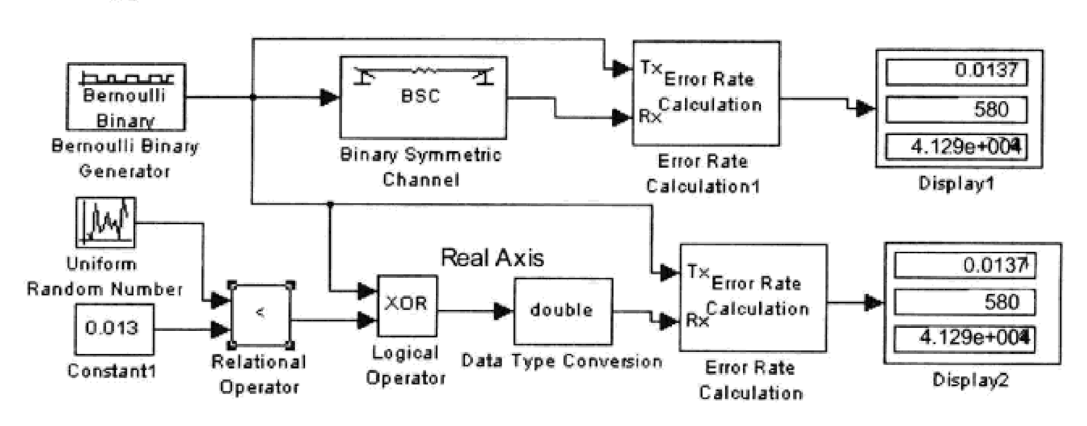


图 9-9 二进制对称信道建模和误码率测试模块的应用测试模型

误码率计算模块 Error Rate Calculation 的可设置参数有：

- 接收延时量，即设置接收数据和发送数据之间的相对时间延迟样值数。
- 计算延时量，即设置开始比较计算时需要忽略的样值数。
- 计算模式，可以是全帧统计误码，也可以对帧中指定位置统计误码，指定帧中误码统计位置的方式可以用掩码（Mask）方式，也可以使用输入端口控制。

误码率计算模块默认将统计结果输出到 MATLAB 工作空间，也可以选择将统计结果输出到端口，这样就可以通过 Display 模块将误码统计结果显示出来。误码统计结果是长度为 3 的向量，其中 3 个元素分别代表误码率、总误码数目和总统计码字数目。例如，图 9-9 中显示了模型执行后显示的 Binary Symmetric Channel 输出误码率为 0.0137，总的统计误码数为 580，总统计码字数为 41290。误码率计算模块还可以设置复位端口，以及仿真停止条件等。通常，当误码数达到 10～100 以上，就可以认为统计误码率足够精确的，因此可以设置统计误码数达到 100 作为仿真停止条件，当然也可以指定总的统计码字数作为仿真停止条件。

9.3 滤波器模型

9.3.1 滤波的相关操作

滤波器是指执行信号处理功能的电子系统，它专门用于去除信号中不想要的成分或者增强所需成分。根据性质，滤波器可以分为非线性的、线性的，时不变的、时变的（自适应的），连续的、离散的（数字的），无限脉冲响应（IIR）的、有限脉冲响应（FIR）的等。线性时不变滤波器就是一个线性时不变系统，在不引起歧义的情况下，我们将线性时不变滤波器简称为滤波器。

一个单输入单输出的滤波器通常用其传递函数或冲激响应来表示。如果滤波器的冲激响应是一个时间连续函数 $h(t)$，那么就称为模拟滤波器，其传递函数用拉普拉斯变换 $H(t)$ 表示。如果滤波器的冲激响应是一个离散时间序列 $h(k)$，我们称其为数字滤波器，其传递函数

以 Z 变换 $H(z)$ 来表示。模拟滤波器可根据传递函数综合出由模拟电路（如电阻、电容、电感组成的无源网络）或由运算放大器组成的有源网络来实现。数字滤波器则通常是以时序数字电路或数字信号处理芯片和软件来实现的。

根据滤波器实现中所使用的综合方法的特征不同，可将其划分为巴特沃斯（Butterworth）型，切比雪夫（Chebyshev）1、2 型，椭圆（Elliptic）型等。

在此只介绍如何使用 MATLAB/Simulink 的函数或模块来设计这些滤波器。不同类型的滤波器设计参数有所不同。

所谓滤波器设计，就是根据设计的滤波器类型和参数计算出满足计算要求的滤波器的最低阶数和相应的 3dB 截止频率（Cutoff Frequencies），然后进一步求出对应的传递函数的分子分母系数。

1. 求滤波器的最小阶和 3dB 截止频率

MATLAB 是提供了 buffer、cheb1ord、cheb2ord、ellipord 这 4 个函数分别设计巴特沃斯型，切比雪夫 1、2 型滤波器以及椭圆型模拟滤波器或数字滤波器。它们的调用格式相同，具体如下：

```
[n,Wn] = buttord(Wp,Ws,Rp,Rs)：巴特沃斯型数字滤波器
[n,Wn] = buttord(Wp,Ws,Rp,Rs,'s')：巴特沃斯型模拟滤波器
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs)：切比雪夫 1 型数字滤波器
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs,'s')：切比雪夫 1 型模拟滤波器
[n,Wp] = cheb2ord(Wp,Ws,Rp,Rs)：切比雪夫 2 型数字滤波器
[n,Wp] = cheb2ord(Wp,Ws,Rp,Rs,'s')：切比雪夫 2 型模拟滤波器
[n,Wp] = ellipord(Wp,Ws,Rp,Rs)：椭圆型数字滤波器
[n,Wp] = ellipord(Wp,Ws,Rp,Rs,'s')：椭圆型模拟滤波器
```

对于数字滤波器设计，输入参数 W_p，W_s 分别为归一化的频率 ω_p 和 ω_s。对于模拟滤波器设计，输入参数 f_p，f_s 是不归一化的，即 f_p 和 f_s。R_p，R_s 是以分贝为单位的通带内波动 R_p 和阻带内最小衰减 R_s。返回值 n 为达到设计指标的最低系统阶数，W_n 为数字滤波器的 3dB 归一化截止频率，f_n 为模拟滤波器的 3dB 截止步骤。

- 低通数字滤波器情况：$W_p<W_s$，通带为 0 到 W_p，阻带为 W_s 到 1。
- 低通模拟滤波器情况：$f_p<f_s$，通带为 0 到 f_p，阻带为 f_s 到无穷。
- 高通数字滤波器情况：$W_p>W_s$，阻带为 0 到 W_s，通带为 W_p 到 1。
- 高通模拟滤波器情况：$f_p>f_s$，阻带为 0 到 f_s，通带为 f_p 到无穷。
- 带通数字滤波器情况：$W_s(1)<W_p(1)<W_p(2)<W_s(2)$，阻带为 0 到 $W_s(1)$以及 $W_s(2)$到 1，通带为 $W_p(1)$到 $W_p(2)$。
- 带通模拟滤波器情况：$f_s(1)<f_p(1)<f_p(2)<f_s(2)$，阻带为 0 到 $f_s(1)$以及 $f_s(2)$到无穷，通带为 $f_p(1)$到 $f_p(2)$。
- 带阻数字滤波器情况：$W_s(1)<W_s(1)<W_s(2)<W_p(2)$，通带为 0 到 $W_p(1)$以及 $W_s(2)$到 1，阻带为 $W_s(1)$到 $W_s(2)$。
- 带阻模拟滤波器情况：$W_p(1)<f_s(1)<f_s(2)<f_p(2)$，通带为 0 到 $f_p(1)$以及 $f_p(2)$到无穷，阻带为 $f_s(1)$到 $f_s(2)$。

2. 系统模型转换

系统模型可以用系统的状态方程描述，对于单输入单输出系统还可用其输入/输出之间

的传递函数来表示，根据传递函数的形式不同，又可以分为分子分母为多项式描述的形式、零极点描述形式、部分分式展开（留数）形式等。为了方便这些等价描述形式之间的转换，MATLAB 提供了丰富的系统模型转换函数，其相应调用格式如下：



[*b*,*a*] = ss2tf(***A***,***B***,***C***,***D***,*iu*)：将由 ***A***、***B***、***C***、***D*** 矩阵确定的状态方程转换为第 *iu* 个输入到输出的传递函数的分子系数向量 *b* 和分母系数向量 *a*。

[***A***,***B***,***C***,***D***] = tf2ss(*b*,*a*)：则将传递函数转换为状态方程。

[*z*,*p*,*k*] = tf2zp(*b*,*a*)：将传递函数转换为零极点形式。

[*b*,*a*] = zp2tf(*z*,*p*,*k*)：则将零极点形式转换为传递函数形式。

[*r*,*p*,*k*] = residue(*b*,*a*)：将传递函数转换为部分分式形式。

[*b*,*a*] = residue(*r*,*p*,*k*)：则将部分分式形式转换为传递函数形式。

3. 线性滤波器特性的图示描述

MATLAB 提供了一系列命令来计算线性系统的时间响应，它们的用法请参考帮助文档，其常用的一些命令如下：

- impulse：计算连续（离散）系统的冲激函数。
- step：计算连续（离散）系统的阶跃响应。
- initial：计算连续（离散）系统的零输入响应。

同样，MATLAB 也提供了计算线性系统的频率响应的命令。命令 freqs 用于计算并画出连续系统的幅频响应和相频响应，其调用格式如下：

```
h = freqs(b,a,w)
[h,w] = freqs(b,a,n)
freqs
```

式中，*b* 为传递函数 *H*(*s*)分子多项式系数向量；*a* 为分母多项式系数向量；*w* 是指定计算频率点序列；返回 *h* 是对应于频率点序列 *w* 的复频率响应；*n* 指定计算频率的指数；如果 *w* 省略则自动选取 200 个频率点作计算，如果无输出变量 *h* 则自动作出幅频响应和相频响应图。

命令 freqz 用于计算并画出离散系统的幅频响应和相频响应，其调用格式如下：

```
[h,w] = freqz(ha)
[h,w] = freqz(ha,n)
freqz(ha)
[h,w] = freqz(hd)
[h,w] = freqz(hd,n)
freqz(hd)
[h,w] = freqz(hm)
[h,w] = freqz(hd,n)
freqz(hd)
```

MATLAB 中使用 grpdelay 来计算离散系统的群时延特性，其调用格式如下：

```
grpdelay(b,a)
[gd,w] = grpdelay(b,a,l)
[gd,f] = grpdelay(b,a,n,fs)
[gd,w] = grpdelay(b,a,n,'whole')
[gd,f] = grpdelay(b,a,n,'whole', fs)
```

```
gd = grpdelay(b,a,w)
gd = grpdelay(b,a,f,fs)
grpdelay(Hd)
```

4．梳状滤波器的设计

利用 MATLAB 滤波器设计工具箱中的梳状滤波器的设计函数 iircomb 可以方便地设计出峰值或谷值滤波器 $H(z)$的分子分母系数。其调用格式如下：

```
[num,den] = iircomb(n,bw)
[num,den] = iircomb(n,bw,ab)
[num,den] = iircomb(…,'type')
```

式中，返回值 num、den 分别为 $H(z)$的分子、分母系数向量；n 为梳状滤波器阶数，在数字归一化频率 0～2π 区间，梳状滤波器开槽数等于 n+1。bw 为滤波器开槽的 ab dB 带宽、默认 ab=－3dB。type 项可以是 notch 或 peak，notch 设计的是开槽型梳状滤波器，peak 设计的是峰值型梳状滤波器。

5．滤波器分析与设计图形界面的使用

MATLAB 专门提供了滤波器设计工具箱，而且还通过图形化界面向用户提供更为方便的滤波器分析和设计工具 FDATool。FDATool 命令将打开滤波器分析和设计界面，如图 9-10 所示。相应的 Simulink 模型是 DSP Blockset 中的 Digital Filter Design 模块，不过 Digital Filter Design 还具有滤波器的实现功能。

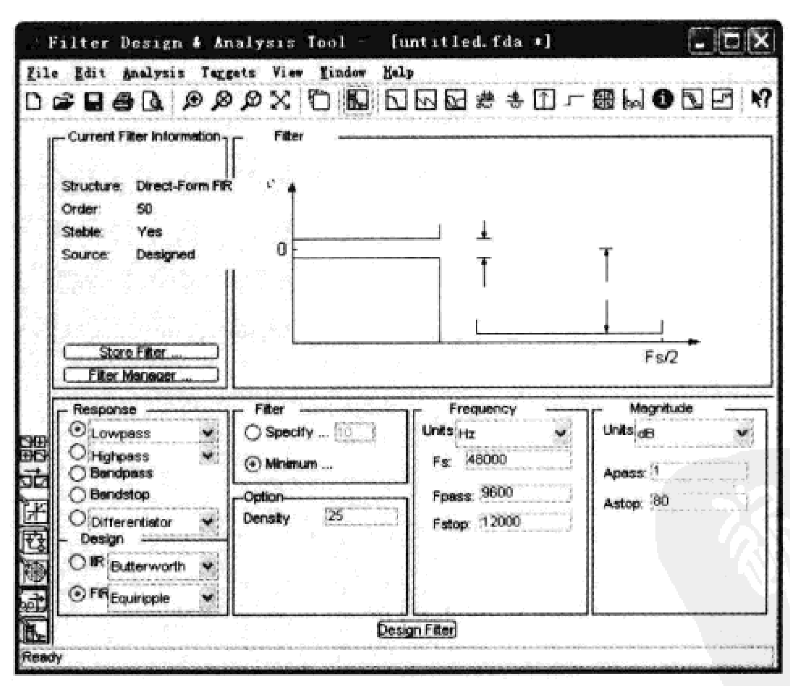


图 9-10　滤波器分析和设计界面

在 FDATool 图形界面下，可以选择滤波器类型、设计模型、滤波器阶数、采样率、通带、阻带频率、幅度特性等一系列参数，然后单击“Design Filter”按钮进行设计运算，通过图形显示滤波器的幅频响应、相频响应、群时延失真、冲激响应、阶跃响应、零极点图、滤波器系数等。Digital Filter Design 模块将实现设计结果。

【例 9-5】 绘制 Chebyshev 1，2 型及 Elliptic 模拟低通滤波器的平方幅频响应曲线，阶数为 2，4，6，8。

其实现的 MATLAB 程序代码如下：

```
% Chebyshev 1 型实现的程序代码如下：
>> n=0:0.01:2;                    %设置频率点
for i=1:4
    switch i
        case 1,
            N=2;
        case 2,
            N=4;
        case 3,
            N=6;
        case 4,
            N=8;
    end

Rp=1;        %设置通带波纹为 1dB
    [z,p,k]=cheb1ap(N,Rp);        %设计 Chebyshev 1 型滤波器
    [b,a]=zp2tf(z,p,k);           %转换为传递函数形式
    [H,w]=freqs(b,a,n);           %求得传递函数的频率特性
    magH2=(abs(H)).^2;            %求得传递函数的幅频特性
    posplot=['2,2,',num2str(i)];
    %将数字 i 转换为字符串,与'2,2,'合并并赋给 posplot
    subplot(posplot);
    plot(w,magH2);
    title(['N=' num2str(N)]);
    %将数字 N 转换为字符串与'N='合并作为标题
    xlabel('w/wc');ylabel('Chebyshev 1 |H(jw)|^2');
    grid on;
end
```

运行程序，输出效果如图 9-11 所示。

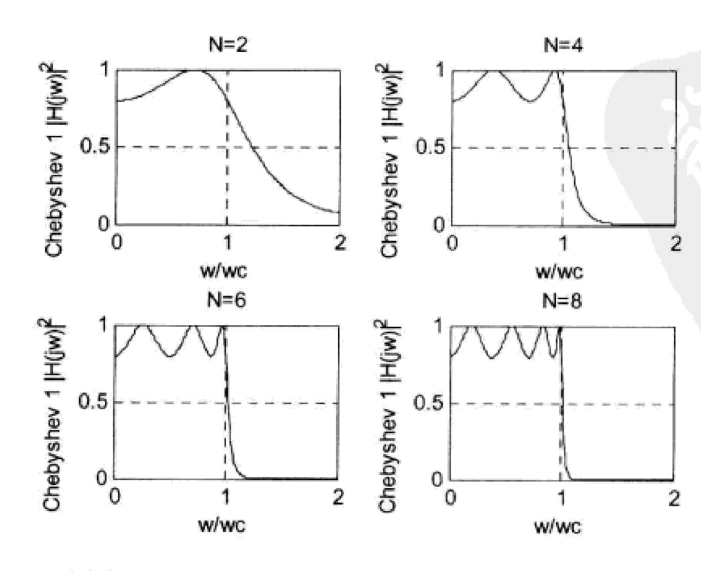


图 9-11 绘制 Chebyshev 1 模拟低通滤波器的平方幅频响应曲线效果



% Chebyshev 2 型实现的程序代码如下：

```
>> n=0:0.01:2;                     %设置频率点
for i=1:4
    switch i
        case 1,
            N=2;
        case 2,
            N=4;
        case 3,
            N=6;
        case 4,
            N=8;
    end
    Rs=16;                         %阻带衰减设置为 16dB
    [z,p,k]=cheb2ap(N,Rs);         %设计 Chebyshev 2 型滤波器
    [b,a]=zp2tf(z,p,k);            %转换为传递函数形式
    [H,w]=freqs(b,a,n);            %求得传递函数的频率特性
    magH2=(abs(H)).^2;             %求得传递函数的幅频特性
    posplot=['2,2,',num2str(i)];
    %将数字 i 转换为字符串,与'2,2,'合并并赋给 posplot
    subplot(posplot);
    plot(w,magH2);
    title(['N=' num2str(N)]);
    %将数字 N 转换为字符串与'N='合并作为标题
    xlabel('w/wc');ylabel('Chebyshev 2 |H(jw)|^2');
    grid on;
end
```

运行程序，输出效果如图 9-12 所示。

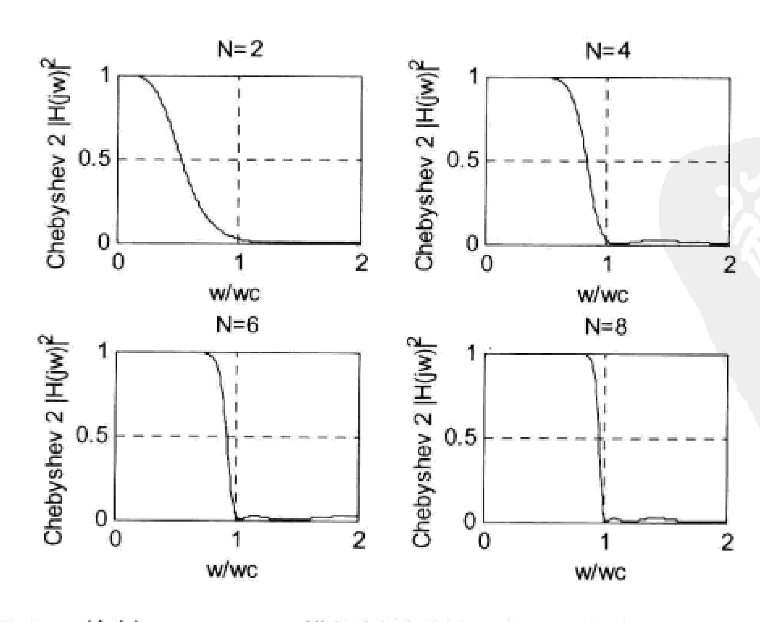


图 9-12　绘制 Chebyshev 2 模拟低通滤波器的平方幅频响应曲线效果

% 绘制 Elliptic 实现的程序代码如下：

```
>> n=0:0.01:2;                          %设置频率点
for i=1:4
    switch i
        case 1,
            N=2;
        case 2,
            N=3;
        case 3,
            N=4;
        case 4,
            N=5;
    end
    Rp=1; Rs=15;                        %设置通带波纹为 1dB,阻带衰减设置为 15dB
    [z,p,k]=ellipap(N,Rp,Rs);           %设计椭圆滤波器
    [b,a]=zp2tf(z,p,k);                 %转换为传递函数形式
    [H,w]=freqs(b,a,n);                 %求得传递函数的频率特性
    magH2=(abs(H)).^2;                  %求得传递函数的幅频特性
    posplot=['2,2,',num2str(i)];
    %将数字 i 转换为字符串,与'2,2,'合并并赋给 posplot
    subplot(posplot);
    plot(w,magH2);
    ylim([0 1]);                        %绘出幅度平方函数
    title(['N=' num2str(N)]);
    %将数字 N 转换为字符串与'N='合并作为标题
    xlabel('w/wc');ylabel('椭圆 |H(jw)|^2');
    grid on;
end
```

运行程序，输出效果如图 9-13 所示。

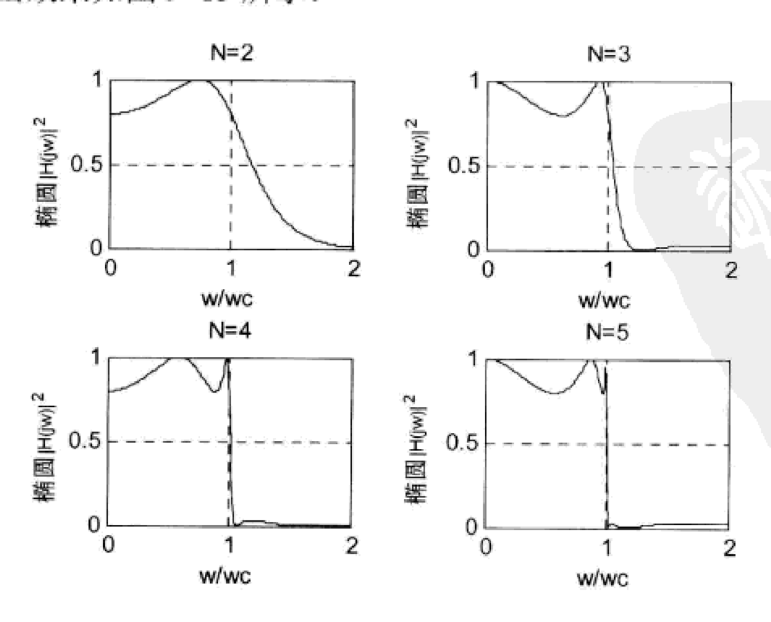


图 9-13 椭圆模拟原形滤波器平方幅频图



【例 9-6】 绘制 Butterworth 低通模拟原型滤波器的幅频平方响应曲线，阶数分别为 2, 5, 10, 20。

其实现 MATLAB 程序代码如下：

```
n=0:0.01:2;                          %频率点
for i=1:4                            %取 4 种滤波器
    switch i
        case 1, N=2;
        case 2; N=5;
        case 3; N=10;
        case 4; N=20;
    end
    [z,p,k]=buttap(N);               %设计 Butterworth 滤波器
    [b,a]=zp2tf(z,p,k);              %将零点极点增益形式转换为传递函数形式
    [H,w]=freqs(b,a,n);              %按 n 指定的频率点给出频率响应
    magH2=(abs(H)).^2;               %给出传递函数幅度平方
    hold on;
    plot(w,magH2);                   %绘制传递函数幅度平方
end
xlabel('w/wc');                      %显示横坐标
ylabel('|H(jw)|^2');                 %显示纵坐标
title('Butterworth 模拟原型滤波器');   %标题显示
text(1.5,0.18,'n=2');                %作必要的标记
text(1.3,0.08,'n=5');
text(1.16,0.08,'n=10');
text(0.93,0.98,'n=20');
grid on;
```

程序运行结果如图 9-14 所示。可以看到，滤波器的幅频平方特性随着频率单调下降。随着滤波器阶数的增大，其幅频特性逐渐接近于矩形，与前面介绍的 Butterworth 滤波器的特性一致。

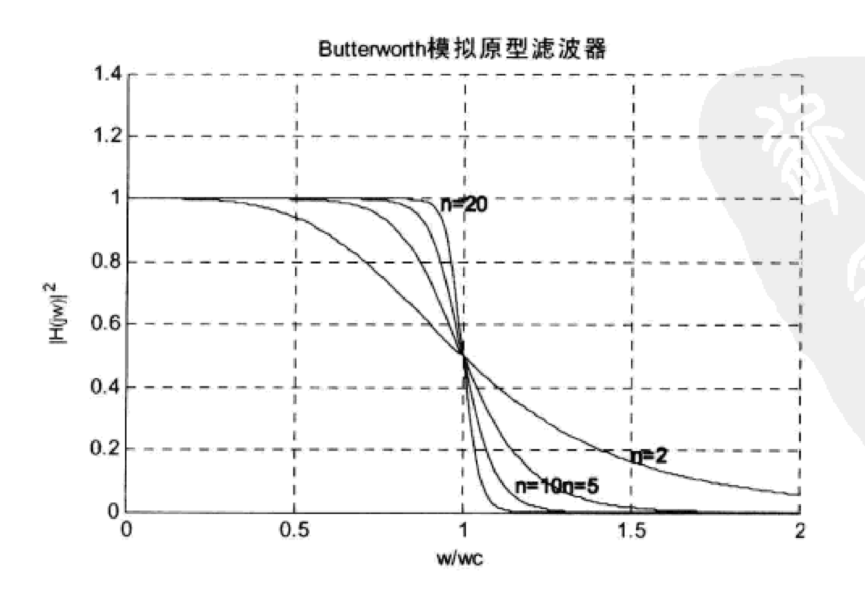


图 9-14 Butterworth 滤波器原型平方幅频

【例 9-7】 在采样率为 600Hz 下设计一个在 60Hz 的地方开槽陷波的梳状滤波器。

其实现的 MATLAB 程序代码如下：

```
>>clear all;
fs = 600; fo = 60;
q = 35; bw = (fo/(fs/2))/q;
[b,a] = iircomb(fs/fo,bw,'notch'); % 开槽型梳状滤波器
fvtool(b,a);
```

运行程序，作出的梳状数字滤波器的效果如图 9-15 所示。

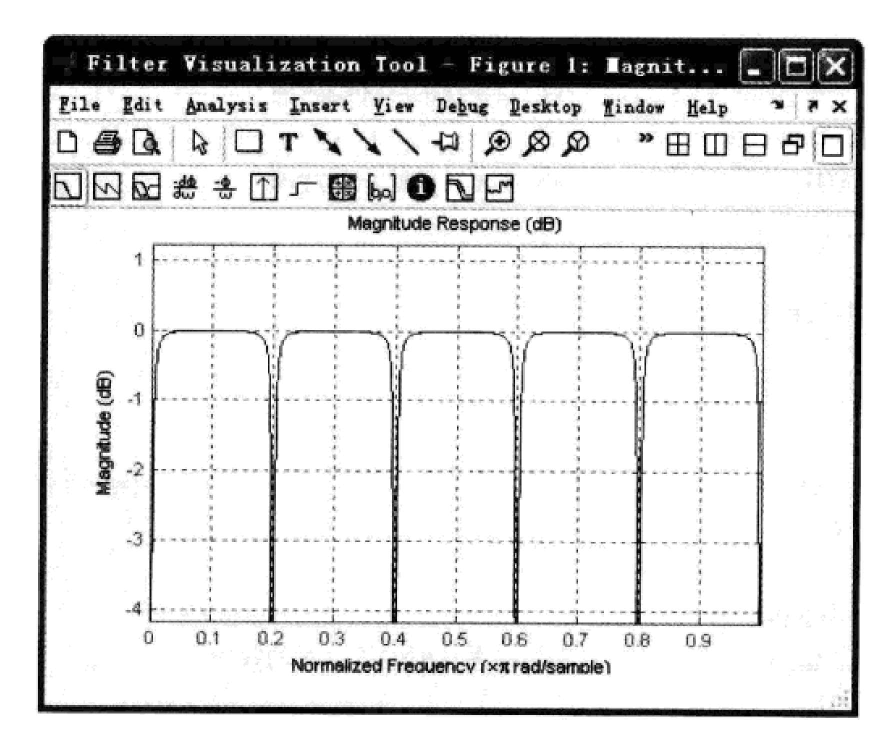


图 9-15 梳状数字滤波器效果



【例 9-8】 设计一个 5 阶的 Chebyshev 1 型带通滤波器，通带波纹 3dB，下边界频率 100Hz，上边界频率 500Hz，绘制幅频响应图。给出该滤波器的脉冲响应、阶跃响应。假定输入 cos(2*pi*30*t)+0.5*sin(2*pi*300*t)+2cos(2*pi*800*t)的信号，求模拟滤波器的输出并给出模拟输入信号和模拟输出信号的 Fourier 振幅谱。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
N=5;Rp=3;                        %滤波器阶数
f1=100; f2=500;                  %滤波器边界频率
w1=2*pi*f1;   w2=2*pi*f2;        %边界频率
[z,p,k]=cheb1ap(N,Rp);           %设计 Chebyshev 1 型原型低通滤波器
[b,a]=zp2tf(z,p,k);              %转换为传递函数形式
Wo=sqrt(w1*w2);                  %中心频率
Bw=w2-w1;                        %频带宽度
[bt,at]=lp2bp(b,a,Wo,Bw);        %频率转换
[h,w]=freqs(bt,at);              %计算复数频率响应
```

```
subplot(2,2,1);
semilogy(w/2/pi,abs(h));                %绘制幅频响应
xlabel('频率/Hz'); title('幅频图');
grid on;
subplot(2,2,2);
plot(w/2/pi,angle(h)*180/pi);           %绘制相频响应
xlabel('频率/Hz'); ylabel('相位/^o'); title('相频图');
grid on;
H=tf(bt,at);                            %在 MATLAB 中表示此滤波器
[h1,t1]=impulse(H);                     %绘制系统的脉冲响应图
subplot(2,2,3);
plot(t1,h1);                            %绘制系统的脉冲响应图
xlabel('时间/s');title('脉冲响应');
[h2,t2]=step(H);                        %绘制系统的阶跃响应图
subplot(2,2,4);
plot(t2,h2);                            %绘制系统的脉冲响应图
xlabel('时间/s');title('阶跃响应');
figure(2);
dt=1/2000;
t=0:dt:0.1;                             %给出模拟滤波器输出的时间范围
u=cos(2*pi*30*t)+0.5+sin(2*pi*300*t)+2*sin(2*pi*800*t);        %模拟输入信号
subplot(2,2,1);
plot(t,u);                              %绘制模拟输入信号
xlabel('时间/s'); title('输入信号');
[ys,ts]=lsim(H,u,t);                    %模拟系统的输入 u 时的输出
subplot(2,2,2);
plot(ts,ys);                            %绘制模拟输出信号
xlabel('时间/s'); title('输出信号');
subplot(2,2,3);
plot((0:length(u)-1)/(length(u)*dt),abs(fft(u))*2/length(u));              %绘输入信号振幅谱
title('输入信号振幅谱');xlabel('频率/Hz');
subplot(2,2,4);
Y=fft(ys);
plot((0:length(Y)-1)/(length(Y)*dt),abs(Y)*2/length(Y));                   %绘制输出信号振幅谱
title('输出信号振幅谱');xlabel('频率/Hz');
```

运行程序，输出结果如图 9-16 和图 9-17 所示。图 9-16 给出了该程序得到的幅频图、相频图、脉冲响应和阶跃响应。幅频图清楚地给出了通带范围和阻带范围。图 9-17 给出了模拟输入、输出信号的时间域波形及振幅谱。输入信号中含有 30Hz、300Hz 和 800Hz 的波，由滤波器的幅频特性可知，30Hz 和 800Hz 全在阻带内，只有 300Hz 的波可通过滤波器。从输入信号和模拟输出信号的时间域波形和振幅谱，可以看出达到了预期的效果。注意，由该滤波器的相频特性可知，该滤波器并不是线性相位。

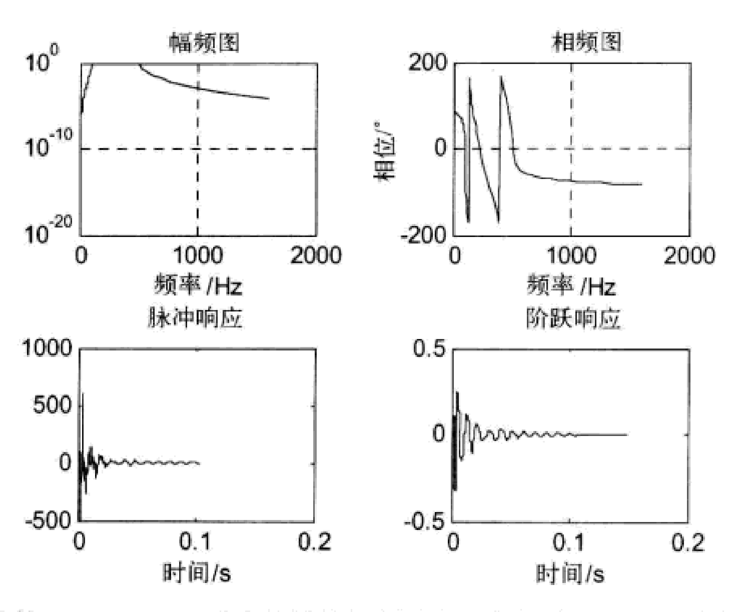


图 9-16　设计的 Chebyshev 1 型滤波器的幅频响应、相频响应、脉冲响应和阶跃响应

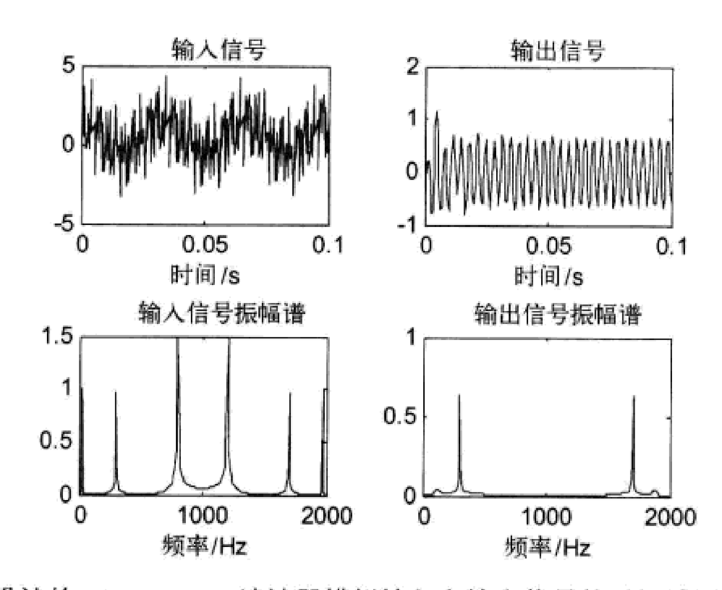


图 9-17　设计的 Chebyshev 1 滤波器模拟输入和输出信号的时间域图形和振幅谱

9.3.2 滤波器的实现分析

根据设计得出的滤波器传递函数，就可以通过建立相应的微分方程或差分方程对其进行实现。所谓实现，就是通过该滤波器对输入信号进行处理，求解其输出信号的过程。因此，对模拟滤波器的实现本质上是求解微分方程的过程，而对数字滤波器的实现则是求解差分方程的过程，这一过程可以通过编程来实现，也可以采用 Simulink 模块方式实现。

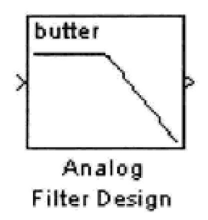
1. 模拟滤波器的实现

根据设计得出模拟滤波器传递函数 $H(s)$，可以写出其状态方程和输出方程，这样通过微分方程的求解命令就可以实现求解过程，也可以将状态空间方程以 S-函数实现，以及通过 Simulink 基本模块库中的连续系统模块来实现。



在 Simulink 的 DSPBlockset 工具箱的 Filter Design 模块库中，还提供了模拟滤波器的设计和实现合成的模块 Analog Filter Design。输入滤波器设计参数，即可通过该模块来实现滤波器。实际上，Analog Filter Design 模块是上述模拟滤波器设计命令的可视化形式，其模块

及参数对话框如图 9-18 所示。



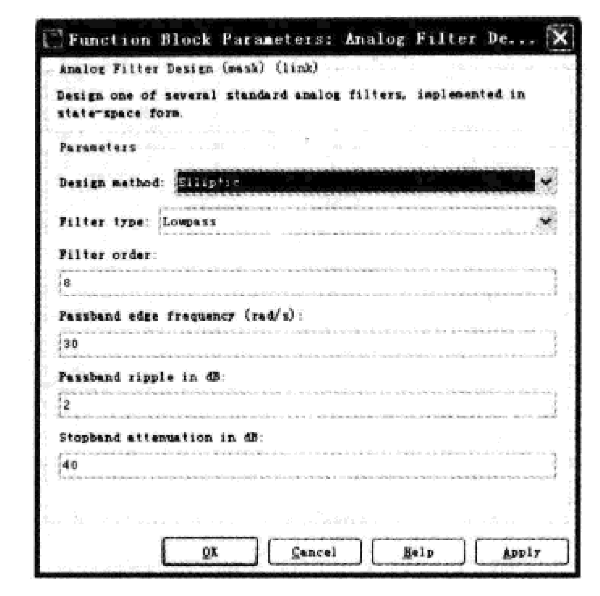


图 9-18　Analog Filter Design 模块及其参数设置对话框

图 9-18 中各项参数的含义与滤波器设计命令中的参数含义一致，它们说明如下：

- Design method（滤波器设计方法）：有 Butterworth（巴特沃斯）、Chebyshe 1（切比雪夫 1 型）、Chebyshe 2（切比雪夫 2 型）和 Elliptic（椭圆型）。
- Filter type（滤波器类型）：Lowpass（低通）、Highpass（高通）、Bandpass（带通）和 Bandstop（带阻）。
- Filter order（滤波器设置阶数）：对于低通和高通滤波器，设置阶数就是滤波器的实现阶数，但是对于带通或带阻滤波器，其实现阶数为设置阶数的 2 倍。
- Lower passband edge frequency（通带下边频率）：单位是 rad/s，是带通和带阻滤波器的设计参数。
- Upper passband edge frequency（通带上边频率）：单位是 rad/s，是带通和带阻滤波器的设计参数。
- Stopband edge frequency（阻带边频率）：单位是切比雪夫 2 型低通和切比雪夫 2 的高通滤波器的设计参数。
- Passband attenuation in dB（阻带内最小衰减）：单位是 dB，是切比雪夫 2 型和椭圆滤波器的设置参数。
- Passband ripple in dB（通带内波动）：单位是 dB，是切比雪夫 1 型和椭圆型滤波器的设计参数。

2. 数字滤波器的实现

用 MATLAB 提供的 filter 函数可以实现数字滤波器，其调用格式如下：

```
y = filter(b,a,X)
[y,zf] = filter(b,a,X)
[y,zf] = filter(b,a,X,zi)
y = filter(b,a,X,zi,dim)
[···] = filter(b,a,X,[],dim)
```

下面参考此函数的用法，如输入以下 MATLAB 代码：

```
>> data = [1:0.2:4]';
windowSize = 5;
filter(ones(1,windowSize)/windowSize,1,data)
```



运行程序，输出结果如下：

```
ans =
    0.2000
    0.4400
    0.7200
    1.0400
    1.4000
    1.6000
    1.8000
    2.0000
    2.2000
    2.4000
    2.6000
    2.8000
    3.0000
    3.2000
    3.4000
    3.6000
```

用 Simulink 基本模块库中的离散系统模块也可以实现数字滤波器，但这些基本模块没有初始状态设置功能。在 DSP Blockset 工具箱中则提供了更为强大且方便的数字滤波器设计和实现工具。其中，Digital Filter Design 模块以图形界面的方式提供了数字滤波器的设计、分析和实现，Digital Filter 模块则提供了以传输函数 $H(z)$为已知条件的多种结构类型数字滤波器的实现，并可设置滤波器的初始状态。

【例 9-9】 已知某数字滤波器的传递函数是：

$$H(z)=\frac{1}{0.1z-0.5}=\frac{10z^{-1}}{1-5z^{-1}}$$

输入信号为 X=[1 0 0 0 0 0 0 0 0 1]，求滤波器的零状态响应输出的解析结果和数值计算结果，并加以对比。

由传递函数 $H(z)$可知，其分子多项式系数向量为 b=[0 10]，分母多项式系数向量为 a=[1 −5]。$H(z)$的反 Z 变换可得到该系统的数字冲激响应序列，即

$$h(k)=a^{k-1}u[k-1]\Leftrightarrow H(z)=\frac{1}{z-a}$$

对比 filter 函数的输出 y 序列和理论计算结果，就可以验证滤波器实现的正确性。由于程序简单，用户在命令窗口直接输入命令验证即可，其程序代码如下：

```
>> clear all;
b=[0 10];
a=[1 -5];                    %滤波器系数
X=[1 0 0 0 0 0 0 0 0 1];     %输入信号序列
```

```
y=filter(b,a,X)                    %滤波
a=5;
k=0:1:9;
hk=a.^(k-1).*(k>0)                 %理论计算H(z)的数字冲激响应
```

运行程序，输出结果如下：

```
y =
     0    10    50   250   1250    6250   31250    156250    781250    3906250
hk =
   0    1    5     25   125   625    3125       15625      78125      390625
```

9.4 调制与解调

调制和解调是互为相反功能的信号频谱搬移过程。在发送端，通过调制将传输信号频谱搬移到指定传输信道的频段上，以便于传输、信道复用以及干扰抑制；在接收端，再以相反的过程——解调一一将传输信号恢复出来。根据调制的性质、被调信号类型可以将调制分为不同的类型。

（1）模拟调制和数字调制

如果被调信号是模拟信号，则相应的调制称为模拟调制方式；反之，如果被调信号携带的是离散的数据符号，则相应的调制方式称为数字调制方式。常见的模拟调制方式有普通调幅（AM）、抑制载波双边带调幅（DSB-SC）、单边带调幅（SSB）、残留边带调幅（VSB）、调频（FM）和调相（PM）等。常见的数字调制方式有幅移键控（ASK）、频移键控（FSK）、相移键控（PSK）以及在这些调制方式基础上的改进调制方式，如正交幅度调制（QAM）、M 元脉冲幅度调制（M-PAM）、差分相位键控（DPSK）、连续相位调制（CPM），最小频移控（MSK）和高斯最小频移键控（GMSK）等。

（2）线性调制和非线性调制

如果调制前后的信号频谱的形状不变，仅发生线性变化，即频谱在频域轴上的位置发生变化，则称为线性调制，如幅度调制、抑制载波双边带调制、单边带调制等。如果在调制过程中，信号频谱不仅发生频率位置的搬移，而且信号频谱的形状还产生了非线性变化，则称为非线性调制。例如，频率调制、相位调制等角度调制方式。

（3）根据调制控制载波不同参数进行分类

一般调制方式通常以正弦信号作为载波，将被调信号（也称为基带信号）搬移到以载波频率为中心频率的频率区域上。调制的方法就是以基带信号去控制载波信号的参数：幅度、角度（频率或相位），以此可将调制分为幅度调制和角度调制（频率调制或相位调制）。

9.4.1 基带模型与调制通带分析

调制输出信号的频谱能量一般集中在调制载波频率附近区域。直接由调制函数建立的仿真模型称为通带（Passband）调制模型。调制载波频率往往很高，在仿真中为了保证信号无失真，必须采用很高的系统仿真采样率，这样仿真步进将不得不设置得非常小，于是系统仿真的计算量和存储数据量大大增加，严重影响仿真执行效率。

改进的方法是将调制信号用等效的复低通信号表示。由于等效复低通信号的最高频率远远小于调制载波频率，相应的系统仿真采样率也就可以大大下降了。以等效复低通信号为基础的系统分析方法就是所谓的复包络方法，相应的调制器等效低通模型称为调制器基带（Baseband）模型。



设任意正弦波调制输出信号为 $x(t)$，用复函数形式表达出来就是：

$$\begin{aligned}x(t) &= r(t)\cos[2\pi f_c t + \phi(t)] \\ &= \mathrm{Re}[r(t)\mathrm{e}^{\mathrm{j}(2\pi f_c t + \phi(t))}] \\ &= \mathrm{Re}[r(t)\mathrm{e}^{\mathrm{j}\phi(t)}\mathrm{e}^{\mathrm{j}2\pi f_c t}] \\ &= \mathrm{Re}[\tilde{x}(t)\mathrm{e}^{\mathrm{j}2\pi f_c t}]\end{aligned} \tag{9-14}$$

式中，$r(t)$ 是幅度调制部分；$\phi(t)$ 是相位调制部分；f_c 是载波频率。复信号

$$\tilde{x}(t) = r(t)\mathrm{e}^{\mathrm{j}\phi(t)} \tag{9-15}$$

包含了与被调信号相关的全部变量，而调制方式的数学性能本质上与载波频率的数值无关，因此具有低通属性的复信号 $\tilde{x}(t)$ 可以用来完全表达调制过程。复信号 $\tilde{x}(t)$ 就称为调制信号 $x(t)$ 的复低通等效信号或调制信号的复包络信号。

9.4.2 解调与模拟调制模型分析

MATLAB 中提供了多个模拟调制解调的模块，下面将做详细的介绍。

1. DSB AM 调制

DSB AM 调制模块对输入信号进行双边带幅度调制。输出为通带表示的调制信号，输入和输出信号都是基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为：$(u(t)+k)\cos(2\pi f_c t + \theta)$。其中，$k$ 为“Input signal offset”参数；f_c 为“Carrier frequency”参数；θ 为“Initial phase”参数。通常设定 k 为输入信号 $u(t)$ 负值部分最小值的绝对值。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的 2 倍。

DSB AM 调制模块及其参数设置对话框如图 9-19 所示。

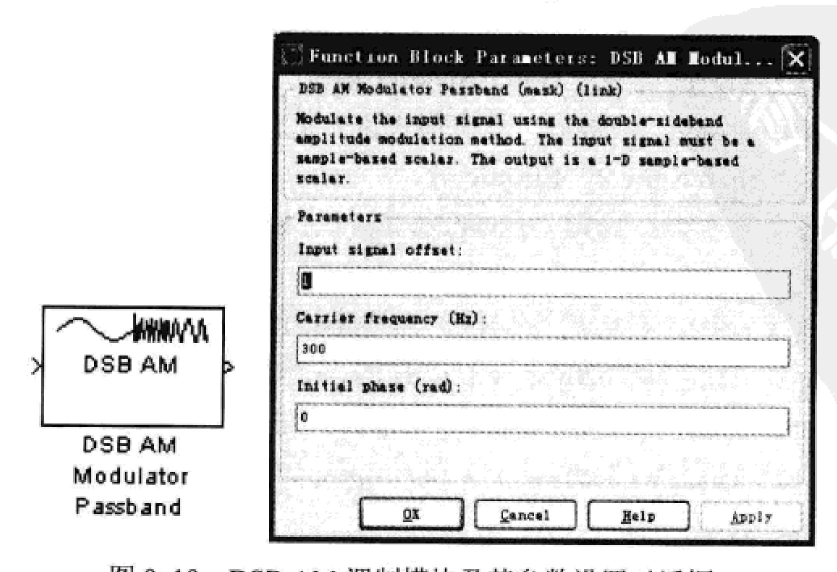


图 9-19　DSB AM 调制模块及其参数设置对话框

DSB AM 调制模块中包含下面几个参数项：

- Input signal offset：设定补偿因子 k，应该大于等于输入信号最小值的绝对值。
- Carrier frequency (Hz)：设定载波频率。
- Initial phase (rad)：设定载波初始相位。

2．DSB AM 解调

DSB AM 解调模块对双边带幅度调制的信号进行解调。输入信号为通带表示的调制信号，且输入/输出信号均为基于采样的实数标量信号。

在解调过程中，DSB AM 解调模块使用了低通滤波器。在通常情况下，"Carrier frequency"参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于"Carrier frequency"参数项的 2 倍。

DSB AM 解调模块及其参数设置对话框如图 9-20 所示。

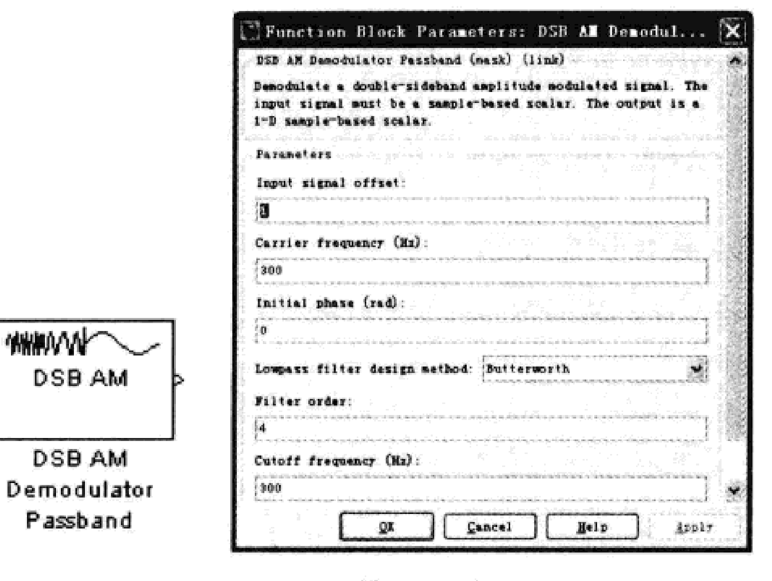


图 9-20　DSB AM 解调模块及参数设置对话框

DSB AM 解调模块中包含下面几个参数项：

- "Input signal offset"文本框：设定输出信号偏移。模块中的所有解调信号都将减去这个偏移量，从而得到输出数据。
- "Carrier frequency (Hz)"文本框：设定调制信号的载波频率。
- "Initial pahse (rad)"：设定发射载波的初始相位。
- "Lowpass filter design method"下拉列表框：滤波器的产生方法，包括 Butterworth、Chebyshev 1、Chebyshev 2、Elliptic 等。
- "Filter order"文本框：设定"Lowpass filter design method"下拉列表框的低通滤波器的截止频率。
- Passband ripple (dB)：设定通带起伏，为通带中的峰-峰起伏。只有当"Lowpass filter design method"选定为 Chebyshev 1 和 Elliptic 滤波器时，本项才有效。
- Stopband ripple (dB)：设定阻带起伏，为阻带中的峰-峰起伏。只有当"Lowpass filter design method"选定为 Chebyshev 2 和 Elliptic 滤波器时，本项才有效。

3．DSBBSC AM 调制

DSBBSC AM 调制模块进行双边带一致载波幅度调制。输出信号为通带形式的调制信

号。输入和输出均为基于采样的实数标量信号。

模块中，如果输入一个时间函数$u(t)$，则输出为：$u(t)\cos(f_c t+\theta)$。其中f_c为“Carrier frequency”参数，θ为“Initial phase”参数。



在通常情况下，“Carrier frequency”参数要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的 2 倍。

DSBBSC AM 调制模块及其参数设置对话框如图 9-21 所示。

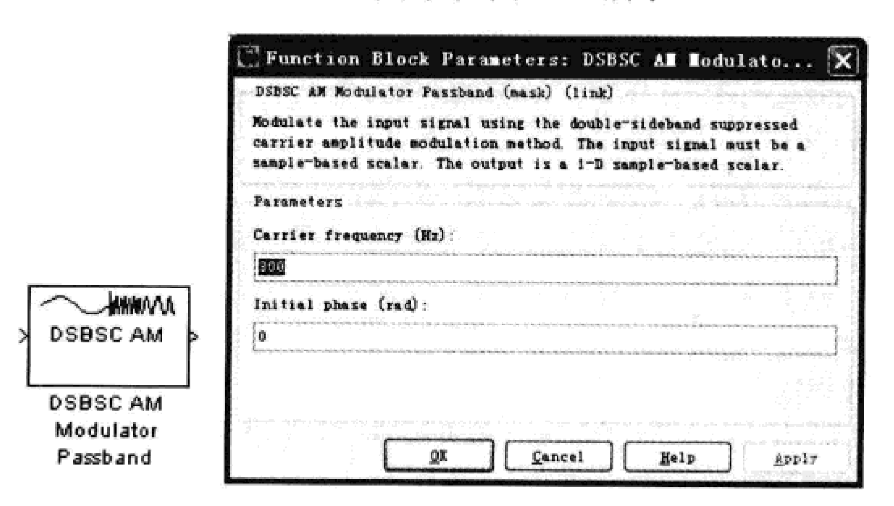


图 9-21 DSBSC AM 调制模块及其参数设置对话框

DSBBSC AM 调制模块中包含下面几个参数项：

- Carrier frequency (Hz)：设定载波频率。
- Initial pahse (rad)：设定初始相位的载波频率。

4．DSBSC AM 解调

DSBSC AM 解调模块对双边带抑制载波幅度调制信号进行解调。输入信号为通带形式的调制信号。输入和输出均为基于采样的实数标量信号。

在通常情况下，“Carrier frequency”参数要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的 2 倍。

DSBSC AM 解调模块及其对数设置对话框如图 9-22 所示。

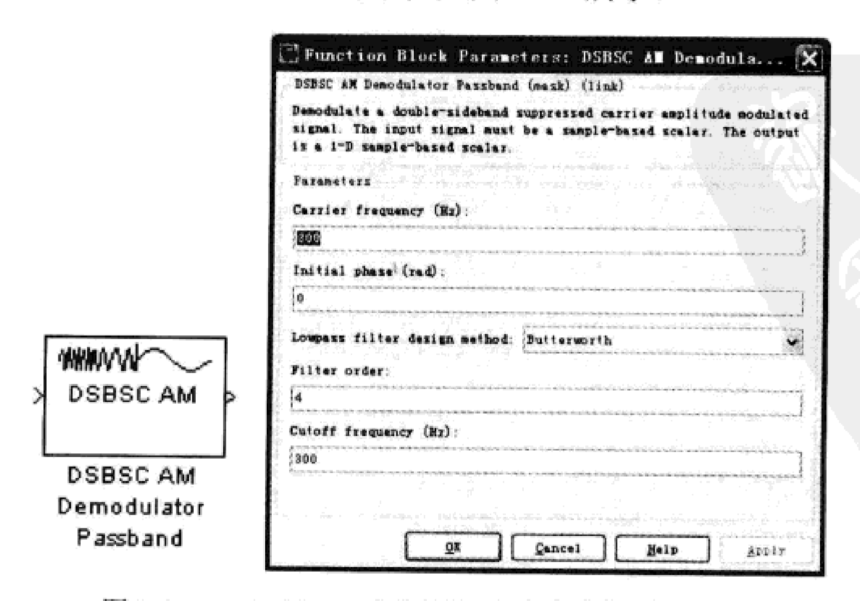


图 9-22 DSBSC AM 解调模块及其参数设置对话框

DSBSC AM 解调模块中包含下面几个参数项：

- Carrier frequency (Hz)：调制信号的载波频率。
- Initial pahse (rad)：设定载波初始相位。
- Lowpass filter design method：滤波器的产生方法，包括 Butterworth、Chebyshev 1、Chebyshev 2、Elliptic 等。
- Filter order：设定“Lowpass filter design method”选项中选定的数字低通滤波器的滤波阶数。
- Cutoff frequency (Hz)：设定“Lowpass filter design method”选项的数字低通滤波器的截止频率。
- Passband ripple (dB)：设定通带起伏，为通带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev 1 和 Elliptic 滤波器时，本项才有效。
- Stopband ripple (dB)：设定阻带起伏，为阻带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev 2 和 Elliptic 滤波器时，本项才有效。

5. SSB AM 调制

SSB AM 调制模块使用希伯特滤波器进行单边带幅度调制。输出为通带形式的调制信号。输入和输出均为基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输入为：$u(t)\cos(f_c t+\theta)\mp u(\hat{t})\sin(f_c t+\theta)$。其中 f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数。$u(\hat{t})$ 表示输入信号的 $u(t)$ 的希伯特转换。式中减号代表上边带，加号代表下边带。

在通常情况下，“Carrier frequency”参数项要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的 2 倍。

SSB AM 调制模块及其对数设置对话框如图 9-23 所示。

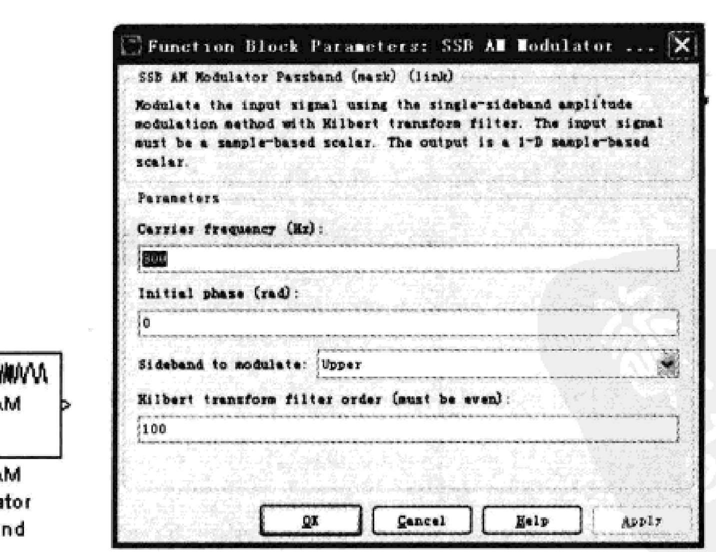


图 9-23 SSB AM 调制模块及参数设置对话框

SSB AM 调制模块中包含以下几个参数项：

- Carrier frequency (Hz)：设定载波频率。
- Initial pahse (rad)：已调制信号的相位补偿 θ。

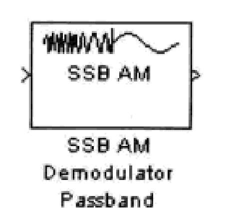
- Sideband to modulate：传输方式设定项，有 upper 和 lower 两种，分别为上边带传输和下边带传输。
- Hilbert Transform filter order：设定用于希伯特转化的 FIR 滤波器的长度。

6．SSB AM 解调

SSB AM 解调模块对单边带幅度调制信号进行解调。输入为通带形式的调制信号。输入和输出均为基于采样的实数标量信号。

SSB AM 解调模块及其参数设置对话框如图 9-24 所示。





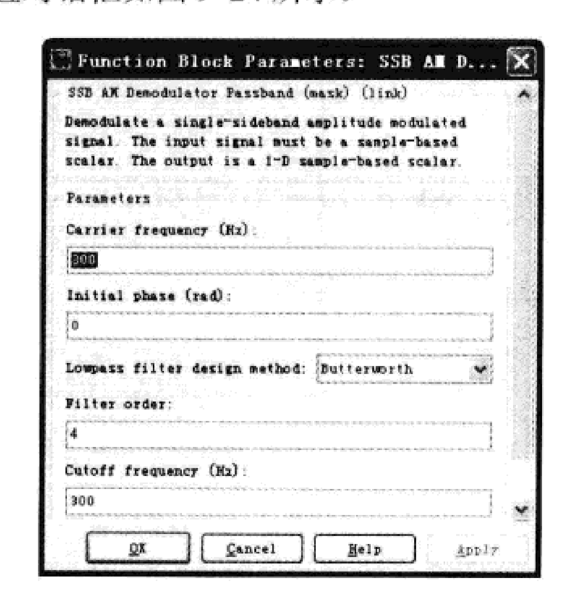


图 9-24　SSB AM 解调模块及参数设置对话框

SSB AM 解调模块包含以下几个参数项：

- “Carrier frequency (Hz)”文本框：调制信号的载波频率。
- “Initial pahse (rad)”文本框：已调制信号的相位补偿 θ。
- “Lowpass filter design method”下拉列表框：滤波器的产生方法，包括 Butterworth、Chebyshev 1、Chebyshev 2、Elliptic 等。
- “Filter order”文本框：设定“Lowpass filter design method”选项中选定的数字低通滤波器的滤波阶数。
- “Cutoff frequency (Hz)”文本框：设定“Lowpass filter design method”选项的数字低通滤波器的截止频率。
- Passband ripple (dB)：设定通带起伏，为通带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev 1 和 Elliptic 滤波器时，本项才有效。
- Stopband ripple (dB)：设定阻带起伏，为阻带中的峰-峰起伏。只有当“Lowpass filter design method”选定为 Chebyshev 2 和 Elliptic 滤波器时，本项才有效。

7．PM 调制

PM 调制模块进行通带相位调制。输出为通带表示的调制信号。输出信号的频率随输入幅度变化而变化。输入和输出信号均采用基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为 $\cos(2\pi f_c t + K_c u(t) + \theta)$。其中 f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数。K_c 为“Modulation constant”参数。

PM 调制模块及其参数设置对话框如图 9-25 所示。

PM 调制模块包含下面几个参数项：

- “Carrier frequency (Hz)”文本框：调制信号的载波频率。
- “Initial pahse (rad)”文本框：表示发射载波的初始相位。
- “Frequency deviation (Hz)”文本框：表示载波频率的频率偏移。

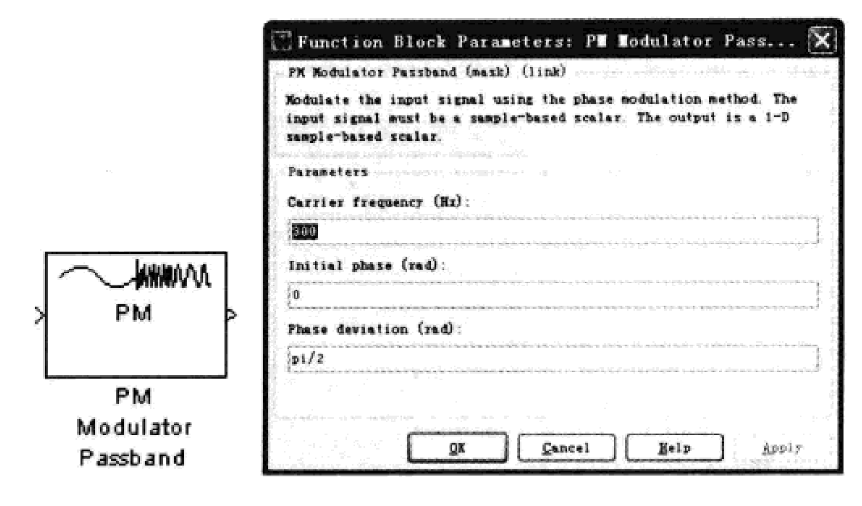


图 9-25　PM 调制模块及其参数设置对话框

8. PM 解调

PM 解调模块对通带相位调制的信号进行解调。输入信号为通带形式的已调信号，输入和输出均为基于采样的实数标量信号。

在解调过程中，模块要使用一个滤波器。为了执行滤波器的希伯特转化，载波频率最好大于输入信号采样时间的 10%。

在通常情况下，“Carrier frequency”参数要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于，“Carrier frequency”参数项的 2 倍。

PM 解调模块及其参数设置对话框如图 9-26 所示。

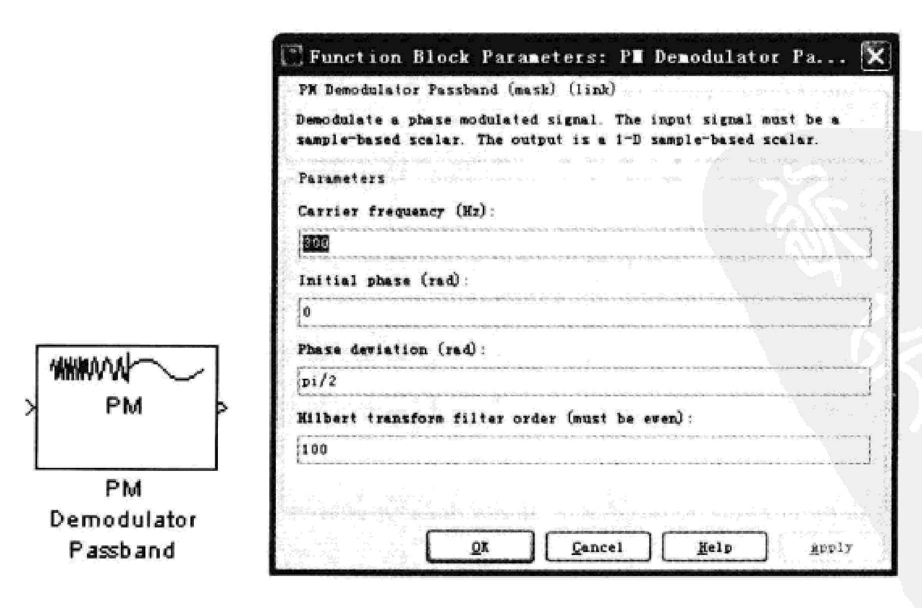


图 9-26　PM 解调模块及其参数设置对话框

PM 解调模块包含以下几个参数项：

- “Carrier frequency (Hz)”文本框：调制信号的载波频率。
- “Initial pahse (rad)”文本框：表示发射载波的初始相位。
- “Phase deviation (Hz)”文本框：表示载波信号的相位偏移。
- “Hilbert transform filter order”文本框：表示用于希伯特转化的 FIR 滤波器的长度。



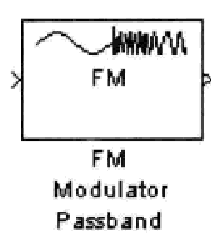
9. FM 调制

FM 调制模块用于频率调制。输出为通带形式的调制信号。输出信号的频率随着输入信号的幅度变化而变化，输入和输出信号均采用基于采样的实数标量信号。

模块中，如果输入一个时间函数 $u(t)$，则输出为：$\cos(2\pi f_c t + 2\pi K_c \int_0^t u(\tau)\mathrm{d}\tau + \theta)$。其中 f_c 为“Carrier frequency”参数，θ 为“Initial phase”参数，K_c 为“Modulation constant”参数。

在通常情况下，“Carrier frequency”参数要比输入信号的最高频率高很多。根据 Nyquist 采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的 2 倍。

FM 调制模块及其参数设置对话框如图 9-27 所示。



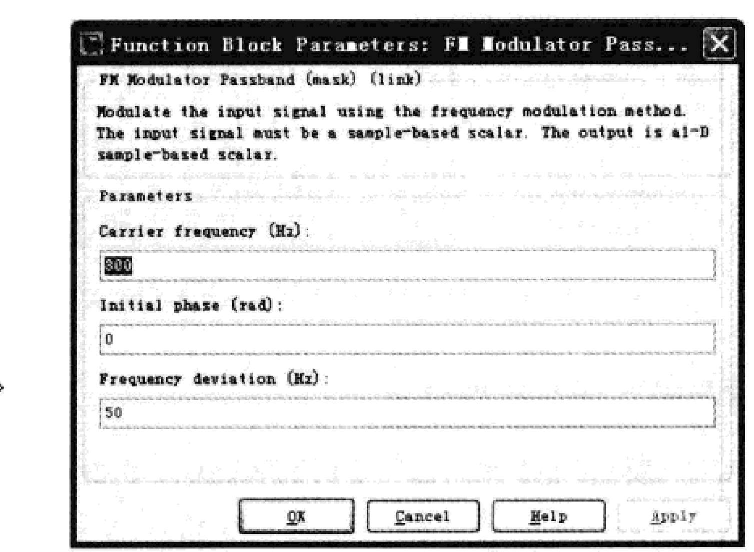


图 9-27 FM 调制模块及其参数设置对话框

FM 调制模块包含以下几个参数项：

- “Carrier frequency (Hz)”文本框：调制信号的载波频率。
- “Initial pahse (rad)”文本框：表示发射载波的初始相位。
- “Frequency deviation (Hz)”文本框：表示载波频率的频率偏移。

10. FM 解调

FM 解调模块对频率调制信号进行解调。输入为通带形式的信号，输入和输出信号均采用基于采样的实数标量信号。

在解调过程中，模块要使用一个滤波器。为了执行滤波器的希伯特转化，载波频率最好大于输入信号采样时间的 10 倍。

在通常情况下，“Carrier frequency”参数要比输入信号的最高频率高很多。根据 Nyquist

采样理论，模型中采样时间的倒数必须大于“Carrier frequency”参数项的 2 倍。

FM 解调模块及其参数设置对话框如图 9-28 所示。

FM 解调模块中包含以下几个参数项：

- “Carrier frequency (Hz)”文本框：调制信号的载波频率。
- “Initial pahse (rad)”文本框：表示发射载波的初始相位。
- “Frequency deviation (Hz)”文本框：表示载波频率的频率偏移。
- “Hilbert transform filter order”（must be even）文本框：表示用于希伯特转化的 FIR 滤波器的长度。

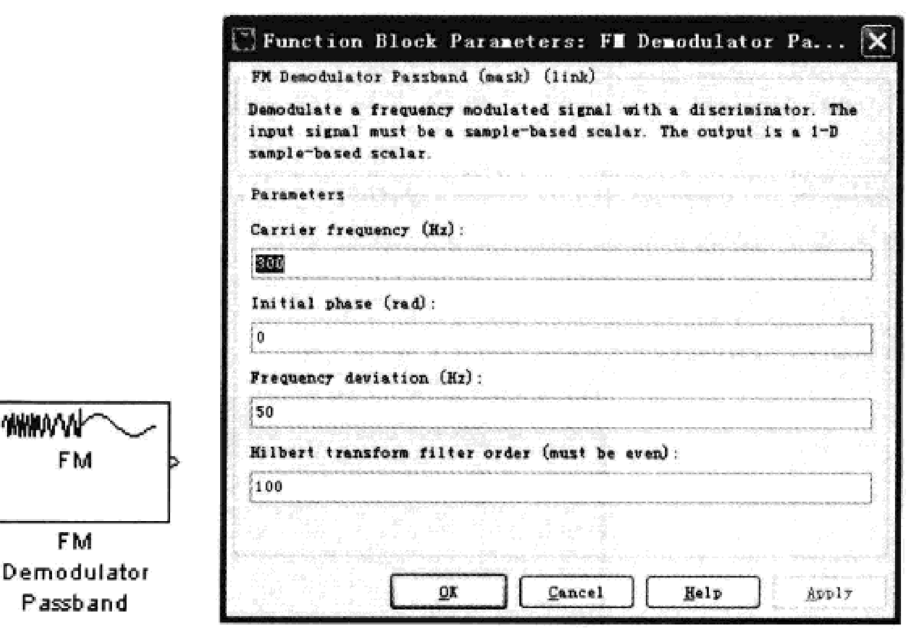


图 9-28　FM 解调模块及其参数设置对话框

9.4.3 解调与数字调制模型分析

MATLAB 中提供了多个数字调制解调的模块，下面将对常用的几个模块做详细的介绍。

1. M-FSK 调制

MATLAB 中提供了 M-FSK Modulator Baseband 模块。该模块进行基带 M 元频移键控制调制。输出为基带形式的已调信号。

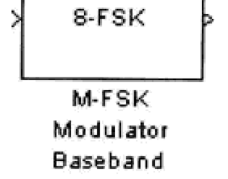
“M-ary number”项参数 M 为已调信号频率。参数“Frequency separation”为已调信号连续频率之间的间隔。

模块的输入和输出为离散信号。“Input type”项决定模块是接收 0 到 M–1 之间的整数，还是二进制形式的整数。

如果“Input type”选项为 Integer，那么模块接收整数输入。输入可以是标量，也可以是基于帧的列向量。如果“Input type”选项为 Bit，那么模块接收 K 的比特的组，称为二进制字。输入可以是长度为 K 的得到或基于帧的列向量（长度为 K 的整数倍）。

M-FSK 调制模块及其参数设置对话框如图 9-29 所示。





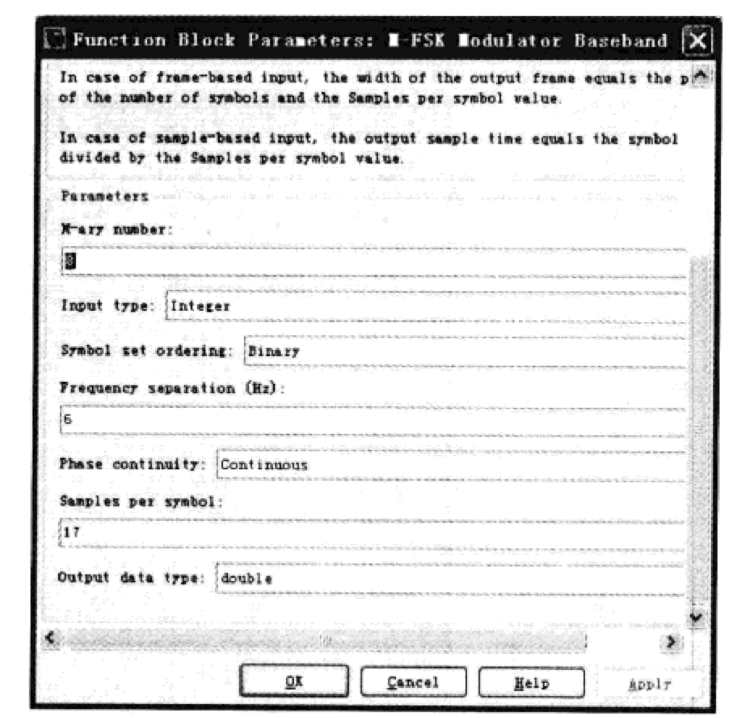


图 9-29 M-FSK 调制模块及其参数设置对话框

M-FSK 调制模块包含以下几个参数项：

- “M-ary number”文本框：表示信号星座图的点数，M 必须为一个偶数。
- “Input type”文本框：表示输入由整数组成还是由位组组成。如果本项设为“Bit”，那么参数 M-ary number 必须为 2^K，K 为正整数。
- “Symbol set ordering”文本框：设定模块如何将每一个输入位组映射到相应的整数。
- “Frequency separation（Hz）”文本框：表示已调信号中相邻频率之间的间隔。
- “Phase continuity”文本框：决定已调制信号的相位是连续的还是非连续的。如果本项设为 Continuous，那么即使频率发生变化，调制信号的相位仍然维持不变。如果该项设为 Discontinuous，那么调制信号由不同频率的 M 正弦曲线部分构成，这样的话如果输入值发生变化，那么调制信号的相位也会发生变化。
- “Samples per symbol”文本框：对应于每个输入的整数或二进制字模块输出的采样个数。
- “Output data type”文本框：设定模块的输出数据类型，可以为 double 或 single。默认为 double 类型。

2. M-FSK 解调

对于 M-FSK Modulator Baseband 模块，MATLAB 提供了 M-FSK Demodulator Baseband 模块，用于基带 M 元频移键控的解调。模块的输入为基带形式的已调制信号。模块的输入和输出均为离散信号。输入可以是标量或基于采样的向量。

“M-ary number”参数 M 为已调信号频率。参数“Frequency separation”为已调信号连续频率之间的间隔。

如果“Input type”选项为 Integer，那么模块输出 0 到 M-1 范围。如果“Input type”选项为 Bit，那么为“M-ary number”选项具有 2^K 的形式，K 为正整数。模块输出 0 到 M-1 之间的二进制形式整数。

M-FSK 解调模块及其参数设置对话框如图 9-29 所示。

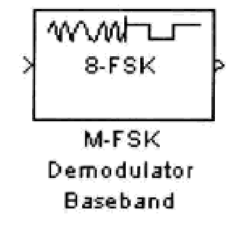


图 9-30　M-FSK 解调模块及其参数设置对话框

M-FSK 解调模块包含以下几个参数项：

- “M-ary number”文本框：表示信号星座图的点数，M 必须为一个偶数。
- “Input type”文本框：表示输入由整数组成还是由位组组成。如果本项设为“Bit”，那么参数 M-ary number 必须为 2^K，K 为正整数。
- “Symbol set ordering”文本框：设定模块如何将每一个输入位组映射到相应的整数。
- “Frequency separation（Hz）”文本框：表示已调信号中相邻频率之间的间隔。
- “Samples per symbol”文本框：对应于每个输入的整数或二进制字模块输出的采样个数。
- “Output data type”文本框：设定模块的输出数据类型，可以为 boolean、int8、uint8、int16、uint16、int32、uint32 或 double。默认为 double 类型。

3. M-PAM 调制

MATLAB 对数字幅度调制提供了 General QAM Modulator Baseband、M-PAM Modulator Baseband、Rectangular QAM Modulator Baseband 等多个模块。下面以 M-PAM Modulator Baseband 模块为例进行介绍。

M-PAM Modulator Baseband 称为 M 相基带幅度调制模块，该模块用于基带 M 元脉冲的幅度调制。模块的输出为基带形式的已调制的信号。模块中“M-ary number”选项的参数 M 为信号的星座图的点数，而且必须是整数。

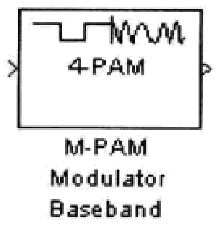
模块使用默认的星座图映射方式，将位于 0～（M-1）的整数 X 映射为复数值[$2X-M+1$]。模块的输入和输出都是离散信号，参数项“Input type”将会决定模块是接收 0～（M-1）的整数，还是接收二进制形式表示的整数。

如果“Input type”设置为 Integer，那么模块接收整数，输入可以是 int8、uint8、int16、uint16、int32、uint32、Single 或 double 类型的基于帧的向量。

如果“Input type”下拉列表框设置为 Bit，那么模块接收 K 位的组，称为二进制字。输入可以是长度为 K 的向量，也可以是长度为 K 的整数倍的基于帧的列向量。在这种情况下，模块可以接受 boolean、int8、uint8、int16、uint16、int32、uint32、Single 或 double 类型的数据。

参数“Constellation ordering”下拉列表框决定模块如何将二进制字分配到信号星座图的点。如果此项设为 Binary，那么模块使用自然二进制编辑星座图。如果此项设置为 Gray，那么模块使用格雷码星座图。

M-PAM 调制模块及其参数设置对话框如图 9-31 所示。



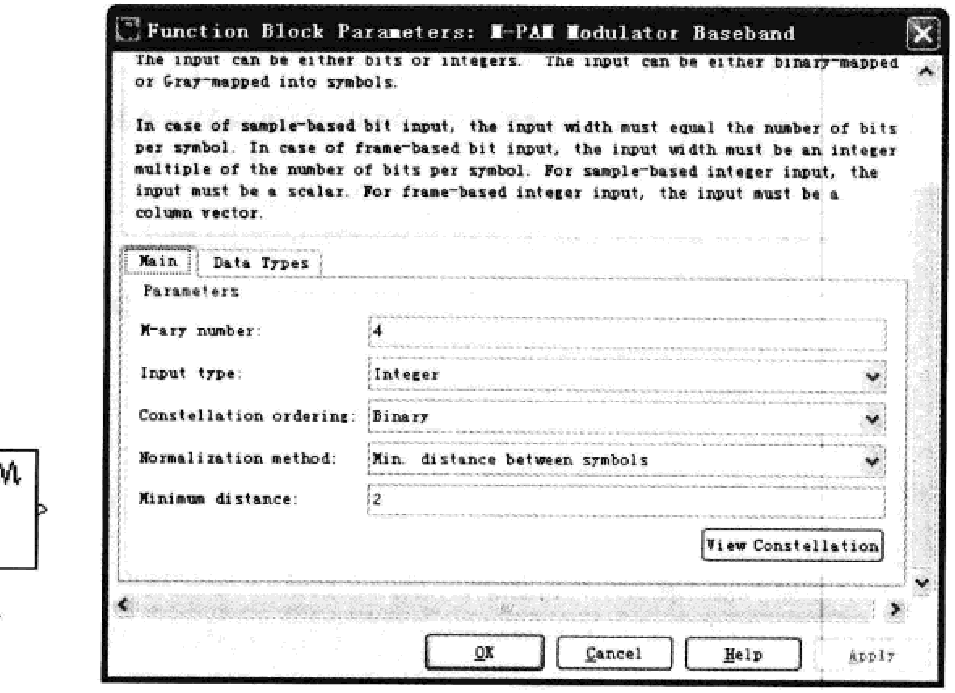


图 9-31 M-PAM 调制模块及其参数设置对话框

如图 9-31 所示，M-PAM 调制模块参数设置对话框中包含“Main”和“Data Type”两个选项卡，默认为“Main”选项卡。下面分别对两个选项卡中的参数项作简单介绍。

（1）Main 选项卡参数说明

- “M-ary number”文本框：表示信号星座图的点数，M 必须为一个偶数。
- “Input type”下拉列表框：表示输入由整数组成还是由位组组成。如果本项设为“Bit”，那么参数 M-ary number 必须为 2^K，K 为正整数。
- “Constellation ordering”下拉列表框：该项决定如何将输入的位组映射成相应的整数。
- “Normalization method”下拉列表框：为一复选框，决定如何测量信号的星座图，有 Min.distance between symbols、Average Power 和 Peak Power 等可选项。
- “Minimum distance”文本框：表示是星座图中两个距离最近点之间的距离。本项只有当“Normalization method”选为“Min.distance between symbols”时才有效。
- Average power (watts)：星座图中符号的平均功率，本项只有当“Normalization method”选为 Average Power 时才有效。
- Peak Power (watts)：星座图中符号的最大功率，本项只有当“Normalization method”

选为 Peak Power 时才有效。

（2）Data Types 选项卡参数项说明

“Data Types”选项卡参数项如图 9-32 所示。

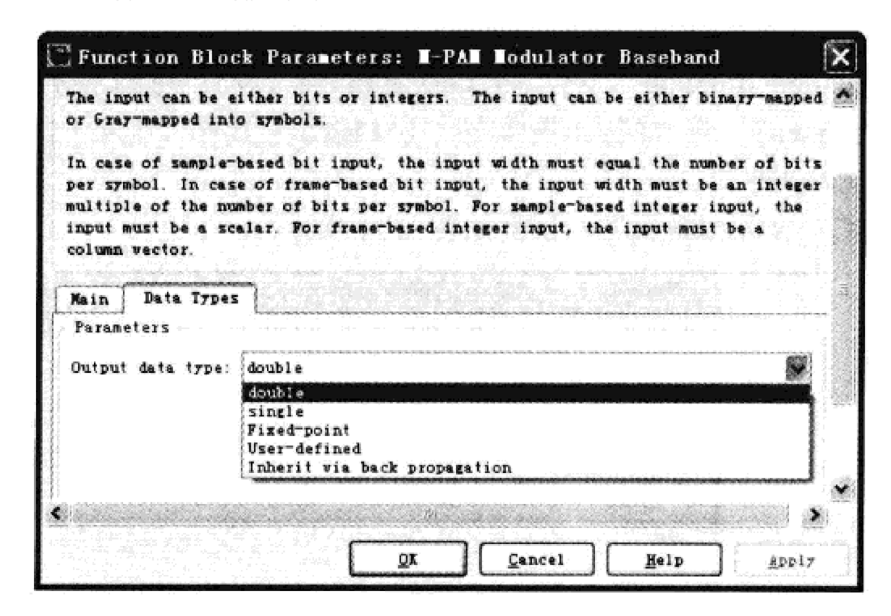


图 9-32 “Data type”类参数项

- “Output data type”下拉列表框：设置输出数据类型。可以设为 double、single、Fixed-point、User-defined 或 Inherit via back propagation 等多种类型。
- Output word length：设定 fixed-point 输出类型的输出字长。本项只有当“Output data type”设为 Fixed-point 时有效并可见。
- User-defined data type：设定带符号的内置或定点数据类型。本项只有当“Output data type”设为 User-defined 时有效并可见。
- Set output fraction legnth to：设置固定点输出比例。本项只有当“Output data type”设为 Fixed-point 或 User-defined 时有效并可见。
- Output fraction length：设置固定点输出数据的分数位数。

4．M-PAM 解调

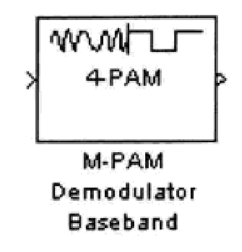
MATLAB 对数字幅度调制提供了 General QAM Demodulator Baseband、M-PAM Demodulator Baseband、Rectangular QAM Demodulator Baseband 等多个模块。下面以 M-PAM Demodulator Baseband 模块为例进行介绍。

M-PAM Demodulator Baseband 称为 M 相基带幅度解调模块，该模块用于基带 M 元脉冲的幅度的解调。模块的输入为基带形式的已调制信号。

参数项“Output type”将会决定模块是产生整数，还是二进制形式表示的整数。如果“Output type”设置为 Integer，那么模块输出整数。如果 “Output type”设置为 Bit，那么模块输出 K 位的组，称为二进制字。参数“Constellation ordering”决定模块如何将二进制字分配到信号星座图的点。

M-PAM 解调模块及其参数设置对话框如图 9-33 所示。

如图 9-33 所示，M-PAM 解调模块参数设置对话框中包含“Main”和“Data Type”两类，默认为“Main”类。下面分别对各类中的参数项简单介绍。



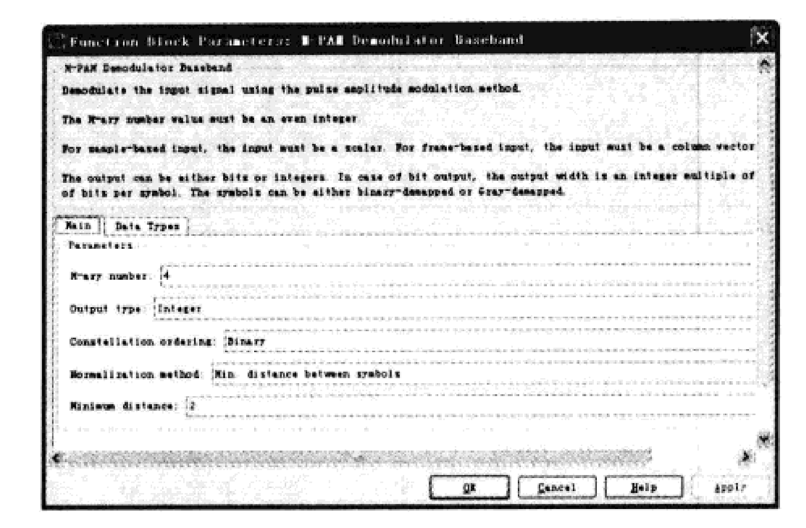


图 9-33　M-PAM 解调模块及其参数设置对话框

（1）Main 选项卡参数说明

- “M-ary number”文本框：表示信号星座图的点数，M 必须为一个偶数。
- “Output type”文本框：表示输出由整数组成还是由位组组成。如果本项设为“Bit”，那么参数 M-ary number 必须为 2^K， K 为正整数。
- “Constellation ordering”文本框：该项决定如何将输入的比特组映射成相应的整数。本项只有在“Output type”设置为 Bit 时有效。
- “Normalization method”文本框：决定如何测量信号的星座图，有 Min.distance between symbols、Average Power 和 Peak Power 等可选项。
- “Minimum distance”文本框：表示是星座图中两个距离最近点之间的距离。本项只有当“Normalization method”选为 Min.distance between symbols 时才有效。
- Average power (watts)：星座图中符号的平均功率，本项只有当“Normalization method”选为 Average Power 时才有效。
- Peak Power (watts)：星座图中符号的最大功率，本项只有当“Normalization method”选为 Peak Power 时才有效。

（2）Data Types 选项卡参数说明

“Data Types”类参数如图 9-34 所示。

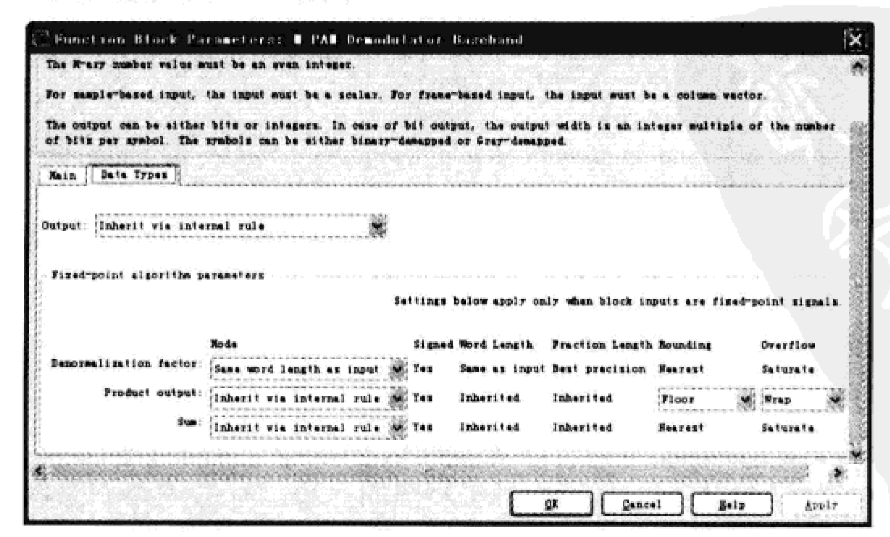


图 9-34　“Data Types”选项卡参数

● “Output”下拉列表框：输出设置项。当参数设置为 Inherit via internal rule（默认）时，模块的输出数据类型由输入端决定。当输入数据为 single 或 double 类型时，输出与输入类型相同。否则输出数据类型将会和本项设置为 Smllest unsigned integer 情况相同。

当参数设置为 Smllest unsigned integer 时，输出数据的类型由模型中结构参数对话框中的“Hardware Implementation”选项决定。如果 Hardware Implementation”选项选择 ASIC/FPGA，输出数据类型为 ideal minimum size。如果 Hardware Implementation”选项选择其他情况时，输出为满足期望最小长度的最小字长无符号整数。

● “Denormalization factor”下拉列表框：可以选定为 Same word length as input 或者 Specify word length，选定后将会出现一个文本框。
● “Product output”下拉列表框：可以选定为 Inherit via internal rule 或者 Specify word length，选定后将会出现一个文本框，具体参见 MATLAB 的联机帮助文档。
● “Sum”下拉列表框：可以选定为 Inherit via internal rule、Same as product output 或者 Specify word length，选定后将会出现一个文本框，具体 MATLAB 的联机帮助文档。

5. M-PSK 调制

MATLAB 中提供了众多的相位调制解调模块，在此我们只对 M-PSK Modulator Baseband 模块作简单介绍。

M-PSK 调制模块进行基带 M 元相移键控制。输出为基带形式的已调信号。“M-ary number”参数 M 表示信号星座图的点数。

M-PSK 调制模块及其参数设置对话框如图 9-35 所示。

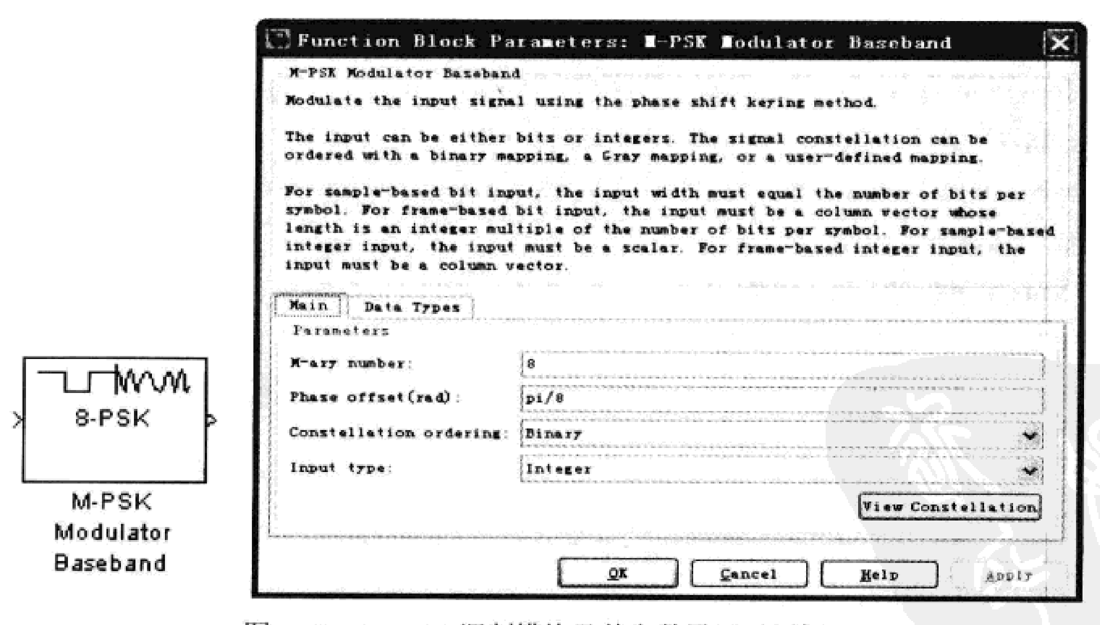


图 9-35 M-PSK 调制模块及其参数设置对话框

如图 9-35 所示，M-PAM 解调模块参数设置对话框中包含“Main”和“Data Types”两个选项卡，默认为“Main”选项卡。下面分别对各类中的参数项简单介绍。

（1）Main 选项卡参数说明

● “M-ary number”文本框：表示信号星座图的点数，M 必须为一个偶数。
● “Phase offset”（rad）”文本框：表示信号星座图中的零点位置。

- "Input type"下拉列表框：表示输入由整数组成还是由位组组成。如果本项设为"Bit"，那么参数 M-ary number 必须为2^K，K为正整数。此时模块的输入信号是一个长度为K的二进制向量，且有$K=\log_2 M$。如果本项为 Integer，那么模块接收范围为[0, M-1]的整数输入。输入可以是标量，也可以是基于帧的列向量。
- "Constellation ordering"下拉列表框：星座图编码方式。如果该项设为 Binary，MATLAB 把输入的K个二进制符号当作一个自然二进制序列；如果该项设为 Gray，MATLAB 把输入的K个二进制符号当作一个 Gray 码。
- "Constelation mapping"下拉列表框：本项只有当"Constellation ordering"项设置为 User-defind 时有效。本项可以是大小为M的行向量或列向量。其中向量的第一个元素对应图中 0+Phase offset 角，后面的元素按照逆时针旋转，最后一个元素对应星座图的点-pi/M+Phase offset。



（2）Data Types 选项卡参数说明

"Data Types"选项卡参数如图 9-36 所示。

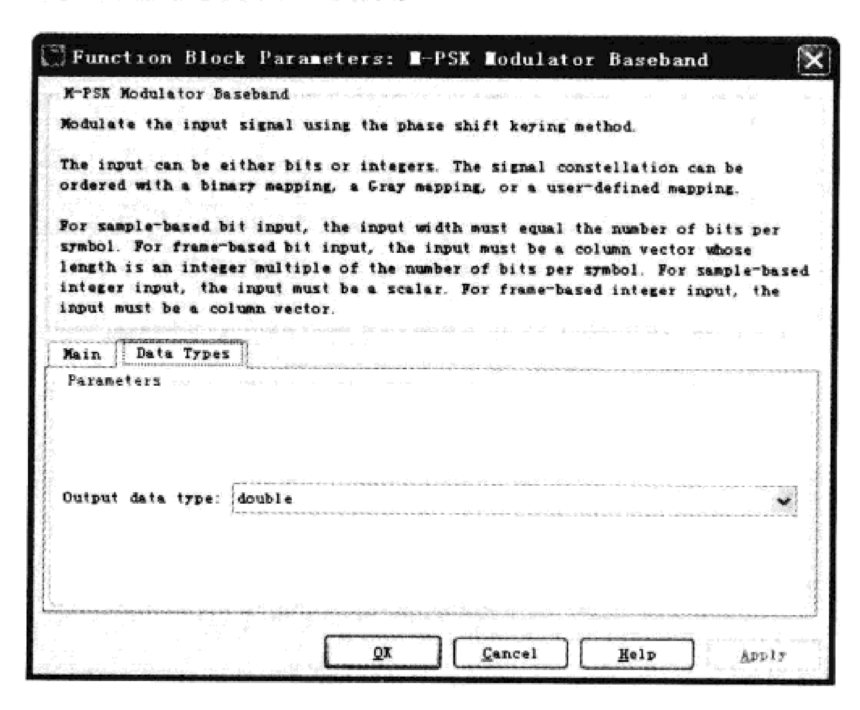


图 9-36 "Data type"选项卡参数

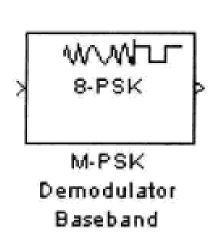
- "Output data type"下拉列表框：设置输出数据类型。可以设为 double、single、Fixed-point、User-defined 或 Inherit via back propagation 等多种类型。
- Output word length：设定输出类型的输出字长。本项只有当"Output data type"设为 Fixed-point 时有效并可见。
- User-defined data type：自定义 d built-in 或 igned fixed-point 数据类型。
- Set output fraction legnth to：设置固定点输出比例。本项只有当"Output data type"设为 Fixed-point 或 User-defined 时有效并可见。
- Output fraction length：设置固定点输出数据的分数位数。

6. M-PSK 解调

对应 M-PSK Modulator Baseband 模块，MATLAB 提供了 M-PSK Demodulator Baseband

模块，用于基带 M 元相移键控调制的解调。输入为基带形式的已调信号。模块的输入和输出都是离散的时间信号。输入可以是标量也可以是基于帧的列向量。参数“M-ary number”表示信号星座图的点数。

M-PSK Demodulator Baseband 模块及其参数设置对话框如图 9-37 所示。



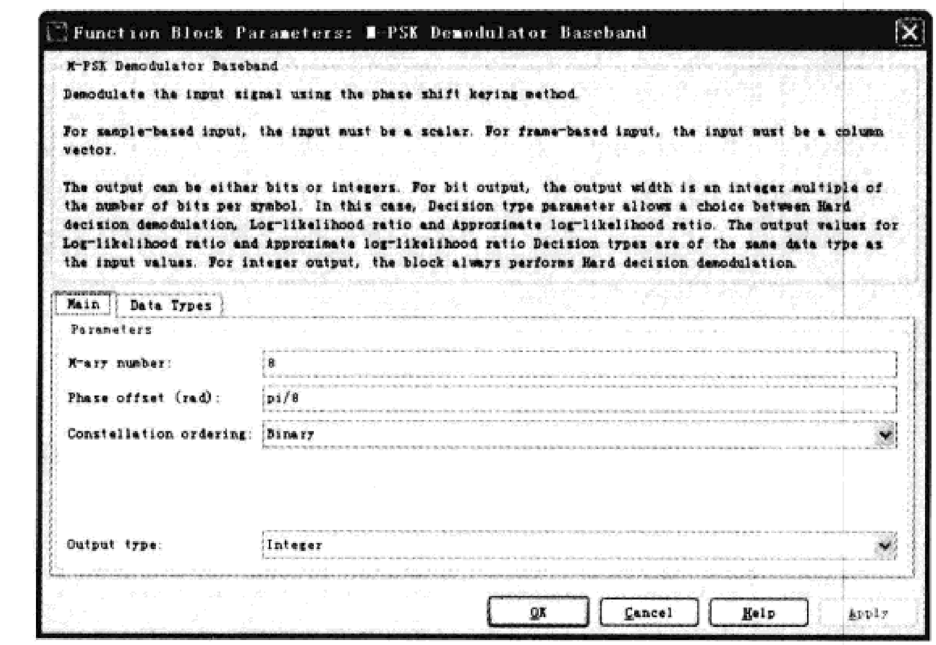


图 9-37　M-PSK 解调模块及其参数设置对话框

如图 9-37 所示，M-PSK 解调模块参数设置对话框中包含“Main”和“Data Type”两个选项卡，默认为“Main”选项卡。下面分别对各类中的参数项简单介绍。

（1）“Main”选项卡参数说明

- “M-ary number”文本框：表示信号星座图的点数，M 必须为一个偶数。
- “Phase offset（rad）”文本框：表示信号星座图中的零点位置。
- “Output type”下拉列表框：表示输出由整数组成还是由位组组成。如果本项设为“Bit”，那么参数 M-ary number 必须为 2^K，K 为正整数。
- “Constellation ordering”下拉列表框：星座图编码方式，决定模块如何将符号映射成输出位或整数。
- “Constellation mapping”下拉列表框：本项只有当“Constellation ordering”选项设置为 User-defined 时有效。本项可以是大小为 M 的行向量或列向量。其中向量的第一个元素对应图中 0 度角，后面的元素按照逆时针旋转，最后一个元素对应星座图的点-pi/M。
- “Decision type”文本框：当“Output type”选定为 Approximate log-likelihood ratio 或者 Log-likelihood ratio 时显示本项。如果选择 Dialog，则在“Noise variance”中输入噪声变化。如果选择 Port，模块中显示用于设定噪声变化的端口。
- “Noise variance”文本框：当“Noise variance source”设定为 Dialog 时显示本项，用于设定噪声变化。

（2）“Data Types”选项卡参数说明

“Data Types”选项卡参数如图 9-38 所示。

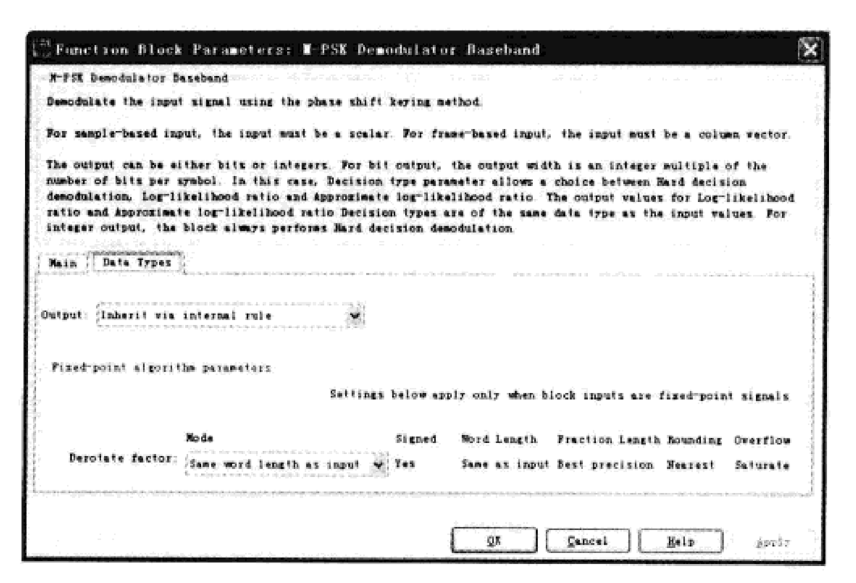


图 9-38 “Data Types”选项卡参数



- “Output”下拉列表框：设定输出。对于位输出，当“Decision type”设置为 Hard decision 时，输出数据类型可以为 Inherit via internal rule、Smallest unsigned integer、double、single、int8、uint8、int16、uint16、int32、uint32、boolean 等类型。对于整数输出，输出数据类型可以是 Inherit via internal rule、Smallest unsigned integer、double、single、int8、uint8、int16、uint16、int32、uint32 类型。

如果本项设置为 Inherit via internal rule（默认项），那么数据的输出类型由输入端决定。如果输入端的输入为 floating-point type 型数据，则输出数据类型相同。如果本项设置为 fixed-point，那么输出数据类型将会和本项设置为 Smallest unsigned integer 时相同。如果本项设置为 Smallest unsigned integer，那么输出数据的类型由模型中结构参数对话框中的“Hardware Implementation”选项决定。如果“Hardware Implementation”选项选为 ASIC/FPGA，并且“Output type”选择为 Bit，那么输出数据类型为 ideal minimum one-bit size。如果“Hardware Implementation”选项选为 ASIC/FPGA，并且“Output type”选择为 Integer，那么输出数据类型为 ideal minimum size。

Derotate factor：本项只使用于“M-ary number”项设为 2、4、8，输入为 fixed-point 类型同时“Phase offset”项为非平凡（即该项当 M=2 时为 π/2 的整数倍；当 M=4 时为 π/4 的奇数倍；当 M=8 时为任意值）的情况。本项有两个可选项：Same word length as input 和 Specify word length。在输出为位的情况下如果“Decision type”设置为 Log-like lihood ratio 或者 Approximate log-likelihood ratio 类型时，输出与输入的数据类型相同。

9.5 模拟线性调制

每一种调制都通过几个特点来表征：

- 调制信号的时域表达式。
- 调制信号的频域表达式。
- 调制信号的带宽。
- 调制信号的功率分布。
- 调制信号的信噪比。

以下两个函数为用户自定义编写的函数，其源代码如下：

```
function [M,m,df]=fftseq(m,tz,df)
fz=1/tz;
if nargin==2                    %判断输入参数的个数是否符合要求
      n1=0;
else
      n1=fz/df;                 %根据参数个数决定是否使用频率缩放
end
n2=length(m);
n=2^(max(nextpow2(n1),nextpow2(n2)));
M=fft(m,n);                     %进行离散傅里叶变换
m=[m,zeros(1,n-n2)];
df=fz/n;

function p=ampower(x)
%此函数用作计算信号功率
p=(norm(x)^2)/length(x);        %计算出信号能量
t0=0.15;
tz=0.001;
m=zeros(1,501);
for i=1:1:125                   %计算第 1 段信号值的功率
      m(i)=i;
end
for i=1:126:1:375               %计算第 2 段信号值的功率
      m(i)=m(125)-i+125;
end
for i=376:1:501                 %计算第 3 段信号值的功率
      m(i)=m(375)+i-375;
end
m=m/1000;                       %功率归一化
n_hat=imag(hilbert(m));
```

9.5.1 常规双边带调幅

1. 原理与分析

在常规双边带调幅中，载波的幅度包络与输入的调制信号成正比，其时域表达式为：

$$S_{AM}(t)=[A_0+f(t)]\cos(\omega_c t+\theta_c)$$

式中，A_0 为外加的直流分量；$f(t)$ 为调制信号，可以是确定性的信号，也可以是随机信号，通常认为其平均值为 0。

注意常规调制必须保证 $(A_0+f(t))$ 的绝对值大于零。ω_c 为载波的角频率，θ_c 为载波的初始相位。如果调制信号为单频余弦波，即

$$f(t)=A_m\cos(\Omega_m t+\theta_m)$$

则 A_0/A_m 称为调制系数。

若记 $F(f)$ 为调制信号的频域表达式，则已调信号的频域表达式为

$$S_{AM}(f)=\frac{A_0}{2}\delta(f-f_c)+\frac{A_0}{2}\delta(f+f_c)+\frac{1}{2}F(f-f_c)+\frac{1}{2}F(f+f_c)$$

从频域表达式可以看出，已调信号的频带宽度是调制信号的频带的两部：$B_T=2\text{W}$。



此种调制方式占用频带较宽。由于被调信号的包络就是调制信号叠加一个直流，因此易于实现峰值包络解调。在频域表达式中可以看出包含有正弦载波分量，即有部分功率耗用在载波上，没有用于信息的传送，从效率上看，常规调幅调制方式较低，但由于其实现和解调都简单，因而得到广泛应用。

2．利用 MATLAB 实现示例

有一有限长度信号 $S(t)$，其表达式为

$$S(t)=\begin{cases} t, & 0<t<\frac{t_0}{4} \\ -t+\frac{t_0}{4}, & \frac{t_0}{4}<t<\frac{3t_0}{4} \\ t-t_0, & \frac{3t_0}{4}<t<t_0 \end{cases}$$

将其调制在载波 $C(t)=\cos 2\pi f_c t$ 上，假设 $t_0=0.5\text{s}$，$f_a=50\text{Hz}$，调制系数为 a=0.8，求出已调制信号的时域表达式及时域波形，未调信号和已调信号的频谱关系图，计算出已调信号和未调信号的功能，并且考虑有噪声的情况，假设信噪比为 10dB，求出噪声功率。

1）已调信号的时域表达式为：

$$M(t)=[1+0.8S(t)/0.125]\cos 2\pi f_c t$$

这里时域表达式给 $S(t)$ 乘以 1/0.125 是因为调制时要进行归一化，将 $S(t)$ 除以最大值，这样就可以保证了调制系数的正确性。

其波形如图 9-39 所示，其实现的 MATLAB 程序代码如下：

```
>> clear all;
tz=0.001;
fz=1/tz;
df=0.2;                          %频率分辨力
snr=10;                          %定义信噪比,用 dB 表示
sn_li=10^(snr/10);               %信噪比的数值
t0=0.5;                          %定义 t0 信号的持续时间的值
fa=50;                           %定义抽样时间
a=0.8;                           %定义调制系数
t=[0:tz:t0];                     %定义出抽样点数据
% 定义信号 m
m=zeros(1,501);
for i=1:1:125,
     m(i)=i;
end
for i=126:1:375,
     m(i)=m(125)-i+125;
end
for i=376:1:501,
     m(i)=m(375)+i-375;
```



```
end
m=m/1000;
c=cos(2*pi*fa.*t);                              %载波信号
m_n=m/max(abs(m));
[M,m,df1]=fftseq(m,tz,df);                      %fftseq 为用户自定义的函数
M=M/fz;                                         %频率缩放,便于作图
f=[0:df1:df1*(length(m)-1)]-fz/2;               %定义频率矢量
u=(1+a*m_n).*c;                                 %将调制信号调制在载波上
[U,u,df1]=fftseq(u,tz,df);                      %对已调信号作傅里叶变换
U=U/fz;                                         %频率缩放
signal_power=ampower(u(1:length(t)));           %ampower 为用户自定义的函数
pmn=ampower(m(1:length(t)))/(max(abs(m)))^2;    %计算调制信号的功率
eta=(a^2*pmn)/(1+a^2*pmn);                      %计算调制效率
noise_power=eta*signal_power/sn_li;             %计算噪声功率
noise_std=sqrt(noise_power);                    %计算标准差
noise=noise_std*randn(1,length(u));             %产生高斯分布的噪声
r=u+noise;                                      %总接收信号
[R,r,df1]=fftseq(r,tz,df);                      %对总信号进行傅里叶变换
R=R/fz;                                         %频率缩放
%以下为结果显示
signal_power                                    %显示信号功率
eta                                             %显示调制效率
subplot(3,1,1);
plot(t,m(1:length(t)))                          %作出调制信号的曲线
axis([0 0.15 -0.1 0.2]);
title('The message signal');
subplot(3,1,2);
plot(t,u(1:length(t)))                          %作出已调信号的曲线
axis([0 0.15 -2.1 2.1]);
title('The modulated signal');
subplot(3,1,3);
plot(t,c(1:length(t)))                          %作出已调信号的曲线
axis([0 0.15 -2.1 2.1]);
title('The carrier');
```

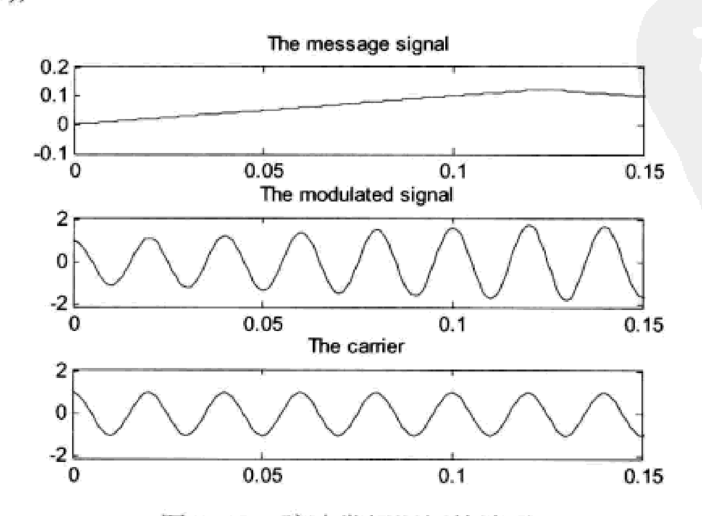


图 9-39 脉冲常规调幅的波形

从图形可以看出，已调信号的包络线就是未调信号，因此可以用峰值包络线进行解调。

2）未调信号的频谱以及已调信号的频谱如图 9-40 所示，紧接以上代码其实现的 MATLAB 程序代码如下：



```
>> figure(2)
subplot(2,1,1)
plot(f,abs(fftshift(M)))     %作出频域的调制信号
xlabel('Freuency'); title('Spectrum of the message signal')
subplot(2,1,2);
plot(f,abs(fftshift(U)))   %作出频域的已调信号
title('Spectrum of the modulated signal');
xlabel('Freuency');
```

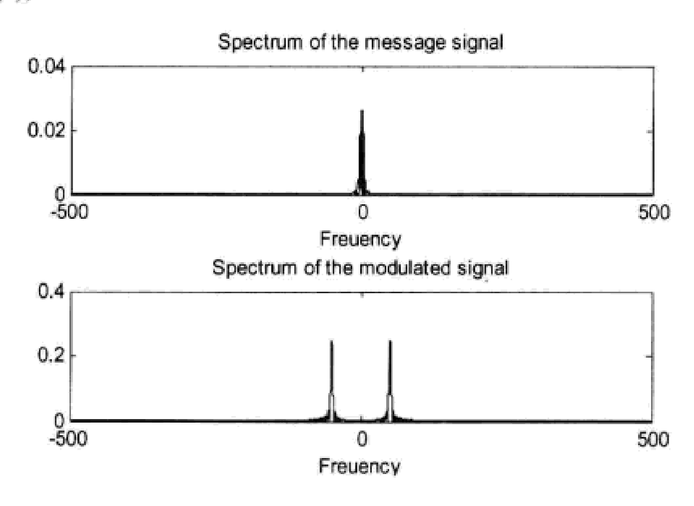


图 9-40　脉冲常规调幅的频谱

从频谱可以看出，原信号的频率分量主要集中在低频但是直流并不多，经调制后，频谱波形搬移到载波的频率附近，并且出现了较多的载频分量，事实上这是两个冲激函数，原因就是调制时采用了给未调信号叠加直流，而直流调频后就成为载频。

3）未调信号的功率 P_S 由下式计算：

$$P_S = \frac{1}{0.5}\left[\int_0^{0.125} t^2 \mathrm{d}t + \int_{0.125}^{0.375} (0.25-t)^2 \mathrm{d}t + \int_{0.375}^{0.5} (t-0.5)^2 \mathrm{d}t\right] = 0.0026$$

归一化频率 P_{S_n} 为

$$P_{S_n} = \frac{P_S}{(0.125)^2} = 0.1650$$

调制效率 η 为

$$\eta = \frac{a^2 P_{S_n}}{1 + a^2 P_{S_n}} = 0.1416$$

可以看出，常规调幅调制的效率是比较代的。

调制信号的功率为

$$P_m = \frac{E[1 + am_n(t)]}{2} = 0.6074$$

在给定的信噪比条件下，可计算噪声功率为

$$P_n = \frac{\eta P_m}{10} = 0.0086$$

在考虑了噪声情况后的噪声时域波形与叠加了噪声的信号的时域波形如图 9-41 所示。紧接以上代码其实现的 MATLAB 代码如下：

```
>> figure(3);
subplot(2,1,1);
plot(t,noise(1:length(t)))   %作出噪声曲线
title('noise sample');xlabel('Time');
subplot(2,1,2);
plot(t,r(1:length(t)));   %作出总信号的时域曲线
title('Signal and noise');xlabel('Time');
```

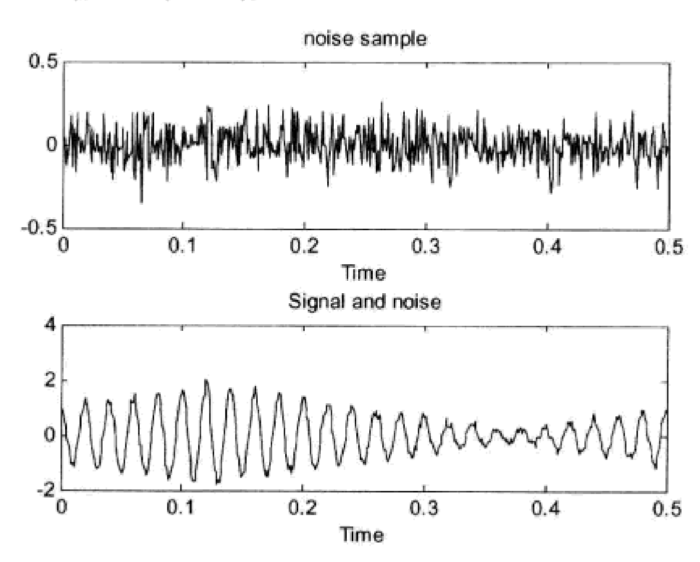


图 9-41　噪声的时域波形与叠加了噪声的已调制信号的时域波表效果

叠加了噪声的信号的频域如图 9-42 所示。紧接以下代码其实现的 MATLAB 程序代码如下：

```
>> figure(4);
plot(f,abs(fftshift(R)));   %作出频域的总信号曲线
title('Signal and noise spectrum');xlabel('Freuency');
```

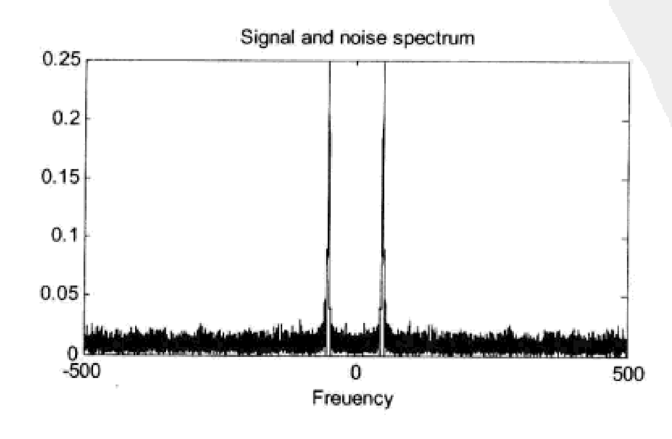


图 9-42　叠加了噪声的已调制信号的频谱



9.5.2 抑制载波双边带调幅

1. 原理与分析

由于常规调幅调制的效率太低，耗用了大量功率，在小功率场合很不方便，而抑制载波双边调幅就克服了效率低的缺点，它的特点是直接将未调信号与载波相乘，而不是先叠加一个直流在未调信号上然后再相乘。时域表达式为：

$$S_{DSB}(t)=Af(t)\cos(\omega_c t+\theta_c)$$

若以单频余弦调制则典型波形如图 9-43 所示。

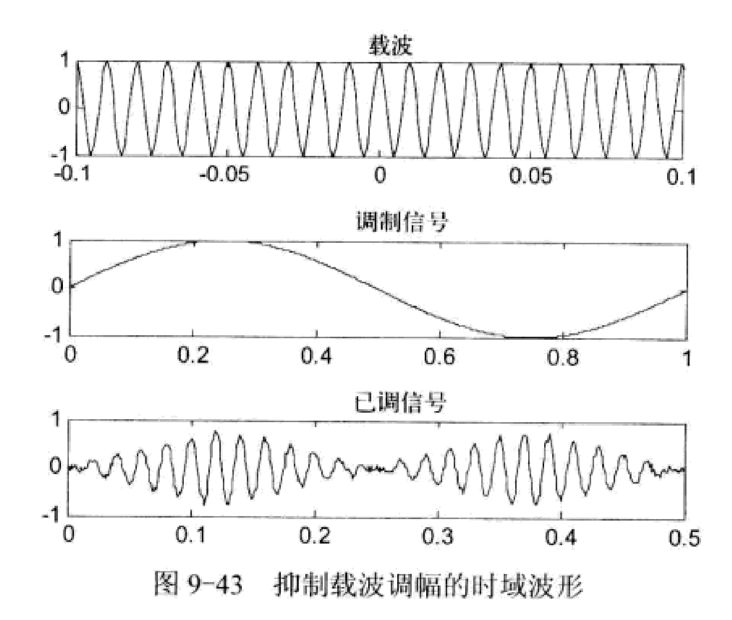


图 9-43　抑制载波调幅的时域波形

抑制载波双边带调制的频谱与常规调幅类似，但没有载频的冲激分量。若记 $F(f)$ 为调制信号的频域表达式，则已调信号的频域表达式为

$$S_{DSB}(f)=\frac{A}{2}F(f-f_c)+\frac{A}{2}F(f+f_c)$$

从频域表达式可以看出，已调信号的频带宽度仍是调制信号的频带的 2 倍：$B_T=2W$。

如对 9.5.2 节例子求其抑制载波的双边带调制，只需要将主文件中的 u 定义改为：

```
u=(a*m_n).*c                    %将调制信号调制在载波上
```

2. 利用 MATLAB 实现示例

下面我们举一个带限信号的抑制载波双边带调幅的例子，未调制信号为：

$$S(t)=\begin{cases}\sin c(200t), & |t|\leqslant t_0\\ 0, & \text{其他}\end{cases}$$

式中，t_0 取 2s；载波为 $C(t)=\cos 2\pi f_c t$，$f_c=100\text{Hz}$，用抑制载波调幅来调制信号，给出调制信号 $M(t)$ 的波形，画出 $S(t)$ 和 $M(t)$ 的频谱。

1）$M(t)=S(t)C(t)$，即

$$M(t)=\begin{cases}3\sin c(10t)\cos(400\pi t), & |t|\leqslant 0.1\\ 0, & \text{其他}\end{cases}$$

其时域波形如图 9-44 所示。其实现的 MATLAB 程序代码如下：

```
clear all;
t0=2;                              %信号持续时间
ts=0.001;                          %抽样时间间隔
fc=100;                            %载波频率
fs=1/ts;
df=0.3;                            %频率分辨力
t=[-t0/2:ts:t0/2];                 %定义时间序列
x=sin(200*t);
m=x./(200*t);
m(1001)=1;                         %避免产生无穷大的值
c=cos(2*pi*fc.*t);                 %载波
u=m.*c;                            %抑制载波调制
[M,m,df1]=fftseq(m,ts,df);         %傅里叶变换
M=M/fs;
[U,u,df1]=fftseq(u,ts,df);         %傅里叶变换
U=U/fs;                            %频率压缩
f=[0:df1:df1*(length(m)-1)];
subplot(2,2,1);
plot(t,m(1:length(t)))             %作出为调信号的波形
axis([-0.4 0.4 -0.5 1.1]);
title('未调信号');xlabel('时间');
subplot(2,2,2);
plot(t,u(1:length(t)))
axis([-0.2 0.2 -1 1.2]);
title('已调信号');xlabel('时间');
subplot(2,1,2);
plot(t,c(1:length(t)))             %作出为调信号的波形
axis([-0.1 0.1 -1 1]);
title('载波');xlabel('时间');
```

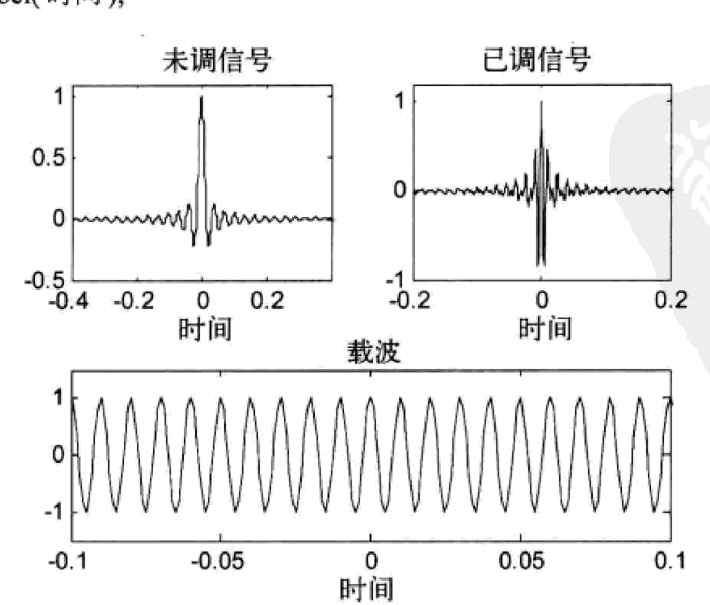


图 9-44　抑制载波调幅波形

2）严格地说，sinc 函数的频谱应该是一个矩形波形的谱，但由于 $S(t)$ 只是 sinc 函数的一段，并且在计算机上使用离散的数字来存储的，导致了计算精度的要求，使频谱结果与矩形波谱不完全一样。频谱如图 9-45 所示，紧接以下的代码其实现的 MATLAB 程序代码如下：



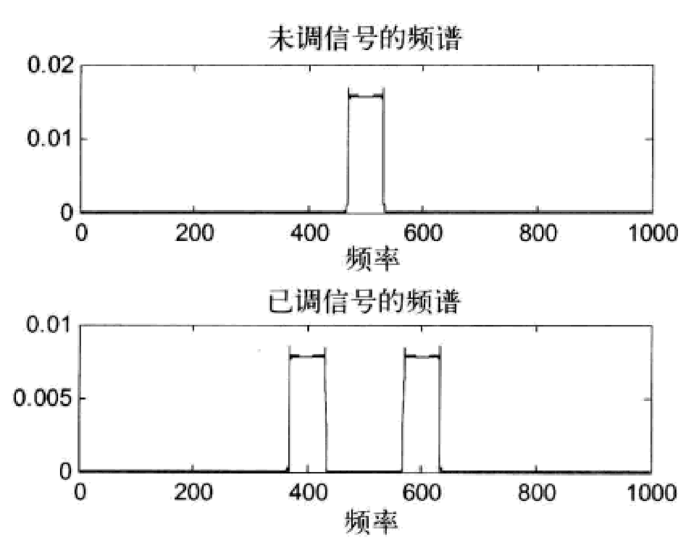


图 9-45　抑制载波调幅频谱

```
figure(2);
subplot(2,1,1);
plot(f,abs(fftshift(M)));
xlabel('频率'); title('未调信号的频谱')
subplot(2,1,2);
plot(f,abs(fftshift(U)))
title('已调信号的频谱');xlabel('频率');
```

9.5.3 单边带调幅

1. 希尔伯特变换

实信号 $x(t)$ 的希尔伯特变换就是将该信号中所有频率成分的信号分量移相 $-\pi/2$ 而得到的新信号，记为 $\hat{x}(t)$ 。对于单频率正弦波信号，设 $m(t)=A\cos(2\pi ft+\phi)$ ，则其希尔伯特变换为：

$$\hat{m}(t)=A\cos\left(2\pi ft+\phi-\frac{\pi}{2}\right)=A\sin\left(2\pi ft+\phi\right) \tag{9-16}$$

对于任意实周期信号 $x(t)$ ，可用周期傅里叶级数展开表示为：

$$x(t)=\sum_{n=0}^{\infty}a_n\cos(2\pi nft+\phi_n) \tag{9-17}$$

其希尔伯特变为：

$$\begin{aligned}\hat{x}(t)&=\sum_{n=0}^{\infty}a_n\cos(2\pi nft+\phi_n-\frac{\pi}{2})\\&=\sum_{n=0}^{\infty}a_n\sin(2\pi nft+\phi_n)\end{aligned} \tag{9-18}$$

实信号 $x(t)$ 的解析信号 $y(t)$ 是一个复信号，其实部为信号 $x(t)$ 本身，虚部为 $x(t)$ 的希尔伯特变换 $\hat{x}(t)$，即

$$y(t) = x(t) + \mathrm{j}\hat{x}(t) \tag{9-19}$$

MATLAB 中提供了希尔伯特变换函数 hilbert 利用 FFT 来计算任意离散时间序列的解析信号序列。其调用语法是：

1）x=hilbert(xr)。xr 是实信号序列，返回 x 是一个复数信号序列；x 的实部就是 xr，x 的虚部则是 xr 的希尔伯特变换序列。

2）x=hilbert(xr, n)。n 作为 FFT 的点数。

例如，对 $x(t) = \sin(t)$ 进行希尔伯特变换的源程序如下：

```
t=0:0.1:30;
y=sin(t);
s_y=hilbert(y); %希尔伯特变换
plot(t,real(s_y),t,imag(s_y),'r:');
legend('原信号','希尔伯特变换结果');
```

程序执行后得出的原信号和希尔伯特变换信号如图 9-46 所示。

Simulink 的通信模块库中也提供了 Analytic Signal 模块来实现对实信号的解析信号计算，其相关用法请读者参考联想帮助文档。

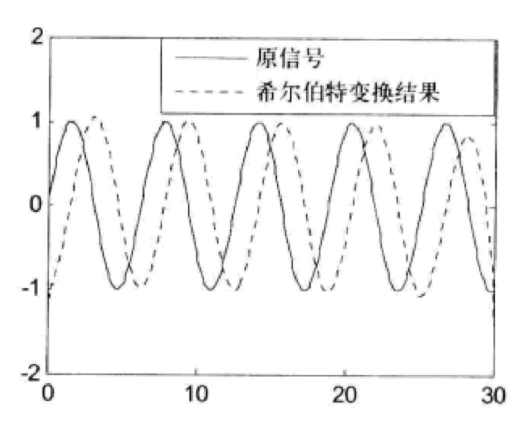


图 9-46 信号 $x(t) = \sin(t)$ 及其希尔伯特变换结果

2. 单边带调幅与解调幅原理

双边带调幅所产生的上下两个边带包含的信息相同，所以只需要传输其中任意一个边带就可以了。将 DSB 信号中的某一边带去除，所得到的就是单边带调制信号。单边带信号的突出优点是节约了传输频带。另外，对于话音信号的单边带解调，可以不用恢复载波相位，甚至接收机的本地载波与发射机的发送载波之间存在少量频率差，话音信号的解调输出失真也不大。

设基带信号以傅里叶级数表示为：

$$m(t) = \sum_{n=0}^{\infty} a_n \cos(2\pi nft + \phi_n) \tag{9-20}$$

则以 $A\cos 2\pi f_c t$ 为载波的双边带输出是：

$$
\begin{aligned}
s_{DSB}(t) &= m(t)A\cos 2\pi f_c t \\
&= A\cos 2\pi f_c t\sum_{n=0}^{\infty} a_n \cos(2\pi nft + \phi_n) \qquad (9\text{-}21) \\
&= \frac{A}{2}\sum_{n=0}^{\infty} a_n \cos(2\pi(f_c + nf)t + \phi_n) + \frac{A}{2}\sum_{n=0}^{\infty} a_n \cos(2\pi(f_c - nf)t - \phi_n)
\end{aligned}
$$

式中，第一项为上边带，第二项为下边带。将第一项从中分离出来，就得到单边带（上边带）的调制输出，即

$$
\begin{aligned}
s_{DSB}(t) &= \frac{A}{2}\sum_{n=0}^{\infty} a_n \cos(2\pi(f_c + nf)t + \phi_n) \\
&= \frac{A}{2}\sum_{n=0}^{\infty} a_n \cos(2\pi nft + \phi_n)\cos 2\pi f_c t - a_n \sin(2\pi nft + \phi_n)\sin 2\pi f_c t \qquad (9\text{-}22) \\
&= \frac{A}{2}m(t)\cos 2\pi f_c t - \frac{A}{2}\hat{m}(t)\sin 2\pi f_c t
\end{aligned}
$$

相应地，下边带调制输出为：

$$s_{DSB}(t) = \frac{A}{2}m(t)\cos 2\pi f_c t + \frac{A}{2}\hat{m}(t)\sin 2\pi f_c t \qquad (9\text{-}23)$$

式中，$\hat{m}(t)$ 为信号 $m(t)$ 的希尔伯特变换。单边带信号的解调方法是相干法，设接收机中本地载波为

$$c(t) = \cos(2\pi(f_c + \Delta f)t + \Delta\phi) \qquad (9\text{-}24)$$

式中，Δf 和 $\Delta\phi$ 分别为本地载波和发送端调制载波之间的频率误差和相位误差。相干解调器的相乘输出信号为：

$$
\begin{aligned}
s_{DSB}(t)c(t) &= \frac{A}{2}\sum_{n=0}^{\infty} a_n \cos(2\pi(f_c + nf)t + \phi_n)\cos(2\pi(f_c + \Delta f)t + \Delta\phi) \\
&= \frac{A}{2}\sum_{n=0}^{\infty} a_n \cos([2\pi(nf - \Delta f)t + (\phi_n - \Delta\phi)]) + \text{高频分量}
\end{aligned} \qquad (9\text{-}25)
$$

经过低通滤波器后，高频分量被滤除，最后得到解调输出为：

$$\tilde{m}(t) = \frac{A}{2}\sum_{n=0}^{\infty} a_n \cos[2\pi(nf - \Delta f)t + (\phi_n - \Delta\phi)] \qquad (9\text{-}26)$$

对比发送基带信号 $m(t)$，解调输出信号中的频率分量存在一定的频率偏移和相位偏移。人耳对于话音波形的相位失真是不敏感的，频率失真会影响到语音音色，但若频率偏移较小（几赫兹到几十赫兹内），对语音的可懂度就不会造成大的影响。在实际的话音单边带通信机中，一般采用一个高稳定度的晶体振荡器或频率合成器来产生本地解调载波，而不需要像双边带的解调那样需要用锁相环（PLL）来恢复载波，这就大大降低了单边带接收机的技术复杂度和成本。

【例 9-10】 设基带信号为一个在 150～400Hz 内、幅度随频率逐渐递减的音频信号，载波信号为 1000Hz 的正弦波，幅度为 1，仿真采样率设为 10000Hz，仿真时间 1s。求 SSB 调制输出信号波形和频谱。

本例采用编程仿真实现，程序代码如下。其中，单边带信号通过式（9-22）和式（9-23）产生，信号的幅度频谱通过 FFT 计算得出。程序代码结果如图 9-47 所示。其中作出了 0～0.01s 内的信号时域波形和 0～2000Hz 内的幅度频谱。由图可知，单边带调制是对基带信号的线性频谱搬移，调制前后频谱仅仅是位置发生变化，频谱形状没有改变。但是，基带信号和单边带调制输出信号时域波形上没有简单的对应关系。

```
clear;
Fs=10000; %仿真的采样率
t=1/Fs:1/Fs:1; %仿真时间点
m_t(Fs*1)=0; %基带信号变量初始化
for f=150:400   %基带信号发生：频率 150～400Hz
    m_t=m_t+0.01*sin(2*pi*f*t)*(400-f);%幅度随线性递减
end
m_t90shift=imag(hilbert(m_t)); %基带信号的希伯尔特变换
carriercos=cos(2*pi*1000*t); %1000 载波 cos
carriersin=sin(2*pi*1000*t); %1000Hz 正交载波 sin
S_SSB1=m_t.*carriercos-m_t90shift.*carriersin;  %上边带 SSB
S_SSB2=m_t.*carriercos+m_t90shift.*carriersin;  %下边带 SSB
```

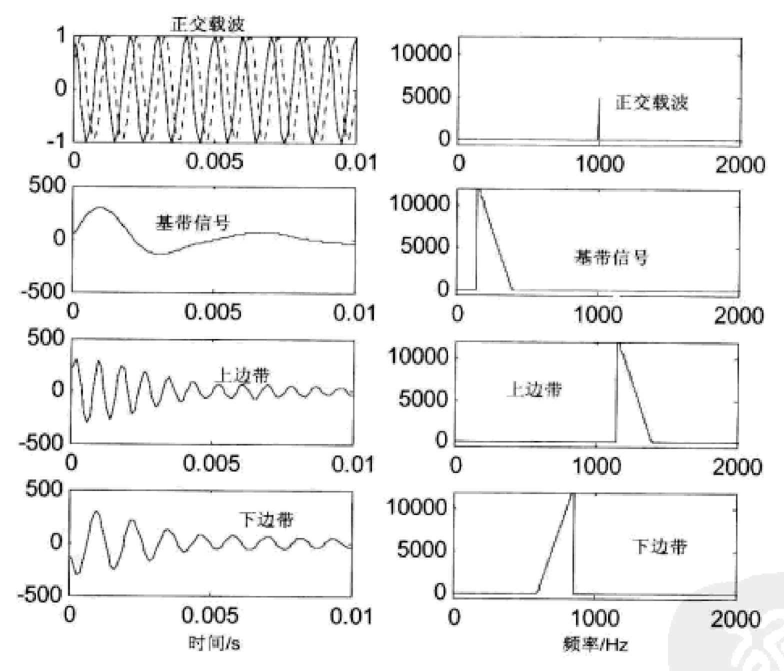


图 9-47　利用希伯尔特变换进行单边带调制的信号波形及对应幅度频谱

```
%下面作出各波形以及频谱
figure;
subplot(421);
plot(t(1:100),carriercos(1:100),t(1:100),carriersin(1:100),':m');%载波
subplot(422);
plot([0:9999],abs(fft(carriercos)));%载波频谱
axis([0 2000 -500 12000]);
subplot(423);
plot(t(1:100),m_t(1:100)); %基带信号
```

```
subplot(424);
plot([0:9999],abs(fft(m_t)));%载波频谱
axis([0 2000 -500 12000]);
subplot(425);
plot(t(1:100),S_SSB1(1:100)); %SSB 波形上边带
subplot(426);
plot([0:9999],abs(fft(S_SSB1))); %SSB 波形上边带
axis([0 2000 -500 12000]);
subplot(427);
plot(t(1:100),S_SSB2(1:100)); %SSB 波形下边带
subplot(428);
plot([0:9999],abs(fft(S_SSB2))); %SSB 波形下边带
axis([0 2000 -500 12000]);
```



9.6 蒙特卡罗仿真的精度分析

9.6.1 蒙特卡罗仿真次数与精度的联系

蒙特卡罗仿真方法本质上是在计算机上进行的随机试验和结果统计分析的过程。试验次数越多，得到的数据样本就越多，那么根据这些样本所得出的统计结果精度和可信程度就越高。

设系统中某事件 A 在一次随机试验中可能发生，也可能不发生，并将其发生概率 $P(A)$作为需要通过仿真来估计的参数，那么可以通过多次独立随机试验，统计这些试验中事件 A 的发生频率，当试验次数足够多时，就可以用频率来近似估计事件发生的概率。

对数据的准确度衡量可以用绝对精度和相对精度两种指标。设数据的准确值（真值）为 x_0，通过仿真得出的估计值 $\hat{x}$，估计值 $\hat{x}$ 一般是一个服从某种分布的随机变量。如果我们有 $1-a$ 的概率确认估计值 $\hat{x}$ 在某一区间 $[x_0-\Delta, x_0+\Delta]$，那么就将概率 $1-a$ 称为置信概率或置信度，即对结果的可信程度，而将区间 $[x_0-\Delta, x_0+\Delta]$ 称为置信区间，将置信区间长度的一半，即 Δ 称为绝对精度，而将绝对精度与真值之比 Δ/x_0 称为相对精度。

在进行仿真时，往往需要根据对仿真结果的精度和置信度要求来确定仿真试验的次数，因为不合理的仿真试验次数会导结果精度过低，或导致过高的计算资源消耗。在使用蒙特卡罗方法进行仿真中的一个重要问题是：给定对仿真结果的置信度以及绝对精度或相对精度指标要求，如何确定所需要的仿真次数。

1.由置信度和绝对精度确定仿真次数

每次蒙特卡罗试验可以看成一次独立的伯努利。例如，通信中传输一个数据符号，可能传输是正确的，也可能是错误的；每次电话拨号，可能被接通，也可能占线；通过随机试验法求圆周率或圆面积时，每次投下的点可能在圆周以内，也可能在圆外等。设一次独立的伯努利试验中事件 A 的概率为 p，那么 n 次独立的伯努利试验的事件发生次数 k 服从二次分布，其可能取值为 $0,1,\cdots,n$，n 次独立效验中事件 A 出现的次数恰为 k 次的概率：

$$P_k(n,p)=\binom{n}{k}p^k(1-p)^{n-k}=\frac{n!}{k!(n-k)!}p^k(1-p)^{n-k} \quad (9\text{-}27)$$

如果以概率 k/n 作概率 p 的估计，设允许绝对误差为 δ，则要求：

$$\left|\frac{k}{n}-p\right|<\delta \tag{9-28}$$

或

$$np-n\delta<k<np+n\delta \tag{9-29}$$

其概率可计算为

$$p_\delta=P(np-n\delta<k<np+n\delta)=\sum_{k=\lceil np-n\delta\rceil}^{\lfloor np+n\delta\rfloor}P_k(n,p) \tag{9-30}$$

因此，给定置信度 p_δ 以及绝对精度 δ，可根据式（9-30）计算出所需要进行仿真的最少次数 n。

然而，这样计算比较复杂，尤其是当需要试验的次数 n 较大时，版式中的组合数计算就难以进行。这种情况下，可通过近似方法进行计算。

根据大数定理，当试验次数 $n\to\infty$，试验中事件发生次数 k 服从均值为 np，方差为 $np(1-p)$ 的正态分布，即

$$P\left(\left|\frac{k}{n}-p\right|<\delta\right)\approx\frac{1}{\sqrt{2\pi}}\int_a^b\exp\left(-\frac{x^2}{2}\right)\mathrm{d}x=\Phi(b)-\Phi(a)=2\Phi(b) \tag{9-31}$$

其中

$$a=\frac{-n\delta}{\sqrt{np(1-p)}},\quad b=\frac{n\delta}{\sqrt{np(1-p)}} \tag{9-32}$$

$\Phi(x)=\frac{1}{\sqrt{2\pi}}\int_0^x\exp\left(-\frac{t^2}{2}\right)\mathrm{d}t=\frac{1}{2}\mathrm{erf}\frac{x}{\sqrt{2}}$ 是拉普拉斯函数。这样，给定置信度 $1-a$ 和绝对精度 δ，以及事件的概率值 p，就可以求解方程：

$$\mathrm{erf}\left(\frac{n\delta}{\sqrt{2np(1-p)}}\right)=1-a \tag{9-33}$$

得出最少仿真次数 n。如果事件的概率值 p 未知，可用估计频率代替。

【例 9-11】已知某通信系统的设计传输错误率为10^{-3}，为了至少有 95%的把握使仿真计算的传输错误率与错误概率真值之间的落差在 2×10^{-4} 范围之内，问至少需要进行多少次仿真（即传输多少个独立符号）？

求解式（9-33）得最少仿真次数为：

$$n=\frac{2p(1-p)}{\delta^2}(\mathrm{erfinv}(1-a))^2 \tag{9-34}$$

式中，erfinv 是误差函数 erf 的反函数。代入题设参数得出最少仿真次数为 95940 次，发现错误数约为 95 个，此时的置信区间为$10^{-3}\pm2\times10^{-4}$。实现的 MATLAB 程序代码如下：

```
>> clear all;
delta=2e-4;                    %绝对误差
p=1e-3;                        %设计误码率
alpha=0.05;                    %显著性水平
```

```
n=floor(2*p*(1-p)/delta^2*(erfinv(1-alpha))^2)
errnum=floor(n*p)
```

运行程序，输出结果如下：

```
n =                              %需要仿真的次数
        95940
errnum =                         %出现错码数
         95
```



除了利用正态分布来近似分析之外，还可以采用更精确的方法：泊松定理指出，在随机试验中事件的发生概率很小，而试验次数很多的情况下，试验中事件的发生次数 k 近似服从参数为 $\lambda = np$ 的泊松分布，即

$$P_k(n,p) \approx \frac{(np)^k}{k!}\exp(-np) \tag{9-35}$$

因此

$$P\left(\left|\frac{k}{n}-p\right|<\delta\right) \approx \sum_{k=\lceil np-n\delta \rceil}^{\lfloor np+n\delta \rfloor} \frac{(np)^k}{k!}\exp(-np) = F(np+n\delta) - F(np-n\delta) \tag{9-36}$$

式中，$F(x)$ 是参数为 λ 的泊松概率分布函数，定义为：

$$F(x) = P(k<x) = \sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!}\exp(-\lambda) \tag{9-37}$$

【例 9-12】在例 9-11 的仿真系统中，设计传输错误率为 10^{-3}，置信区间为 $10^{-3} \pm 2\times10^{-4}$，总独立传输符号为 95940 次，问对仿真结果的置信度可达到多少（分别用泊松分布和正态分布对之进行近似）？

根据上述原理实现的 MATLAB 程序代码如下：

```
>> clear all;
delta=2e-4;                      %绝对误差
p=1e-3;                          %设计误码率
n=95940;                         %仿真次数
P_d_p=poisscdf(n*p+n*delta,n*p)-poisscdf(n*p-n*delta,n*p)
p_d_n=normcdf(n*p+n*delta,n*p,sqrt(n*p*(1-p)))-normcdf(n*p-n*delta,n*p,sqrt(n*p*(1-p)))
```

运行程序，输出结果如下：

```
P_d_p =
         0.9538
p_d_n =
        0.9500
```

2. 由置信度和相对精度确定仿真次数

问题同前，但这里给定仿真的相对精度要求 $r = \delta/p$，则 $\delta = pr$，将这代入式（9-34）得到相对精度下的最小仿真次数为：

$$n = \frac{2(1-p)}{pr^2}(\text{erfinv}(1-a))^2 \tag{9-38}$$

如果给定仿真次数和置信度，则仿真结果的相对精度也可计算出来：

$$r=\sqrt{\frac{2(1-p)}{pn}}\operatorname{erfinv}(1-a) \tag{9-39}$$

注意，当概率 p 很小（例如，对通信传输误码率的仿真情况）时，式（9-39）近似为：

$$r\approx\sqrt{\frac{2}{pn}}\operatorname{erfinv}(1-a) \tag{9-40}$$

式中，pn 的物理意义是 n 次试验中事件出现的平均次数（例如，传输 n 的独立符号后观察到的平均误码出现次数）。在统计误码率时，出现的误码数越多，则统计结果的相对精度就越高。对应于相对精度的置信区间为 $[p(1-r),p(1+r)]$。

【例 9-13】试根据式（9-40）画出置信度为 89%、94%和 98%条件下试验中事件发生次数 pn 与相对精度 r 之间的关系曲线。

其实现的 MATLAB 程序代码如下：

```
>> clear all;
alpha=[0.11 0.06 0.02];
pn=[1 10 100 1000 10000 100000]';
for i=1:1:3
    r(:,i)=sqrt(2./pn).*erfinv(1-alpha(i));
end
loglog(pn,r,'-p');
legend('\alpha=0.11','\alpha=0.06','\alpha=0.02');
xlabel('多次试验中事件发生的次数 np');
ylabel('相对精度 r');
```

运行程序，效果如图 9-48 所示。由图可知，如果要求试验结果的相对精度提高，那么就要使试验中观察到事件发生的次数呈平方数量级增加。在事件发生概率较小的情况下（如对传输错误率的仿真中），将导致总试验次数过分增多，这种情况下蒙特卡罗法的效率将严重下降。

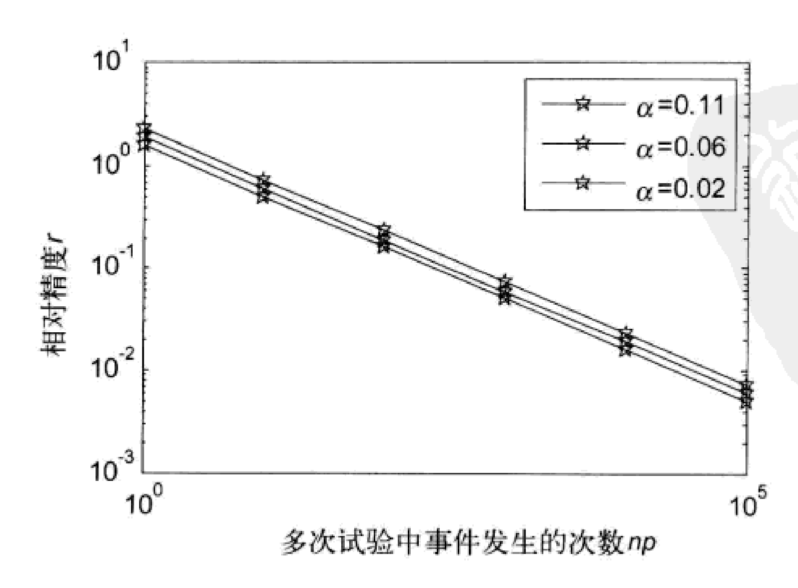


图 9-48　在不同置信度间事件发生次数与相对精度间的曲线关系效果

9.6.2 蒙特卡罗仿真次数的算法

设一次伯努利试验中事件 A 发生的概率为 p，随机变量 X 的取值依试验中事件 A 发生与否而取 1 或 0。那么，其均值和方差为：

$$E(X)=p \tag{9-41}$$

$$\mathrm{Var}(X)=p(1-p) \tag{9-42}$$

如果将 n 次独立的伯努利试验视为一次蒙特卡罗试验，并将其中事件 A 的发生频率作为试验结果，则试验结果是一个随机变量 $Y=\sum_{i=1}^{n}X_i/n$，其均值和方差为：

$$E(Y)=p \tag{9-43}$$

$$\mathrm{Var}(Y)=\frac{\mathrm{Var}(X)}{n}=\frac{p(1-p)}{n} \tag{9-44}$$

通常，一次蒙特卡罗试验所得出的试验结果样本 Y 的方差可以计算出来或由试验样本估计出来。那么，如何在给定仿真精度要求和置信度要求的情况下确定仿真所需的最小次数呢？当一次蒙特卡罗试验中含有的独立伯努利试验次数 n 足够大时，根据大数定理，其输出的试验结果样本 Y 可认为服从正态分布。

设 N 次蒙特卡罗试验所得出的试验结果样本是 $\{y_1,y_2,\cdots,y_N\}$，则根据这 N 个样本对随机变量 Y 的均值估计问题是一个关于正态分布的期望区间估计问题，由 t 分布的分位点的对称性质可得，给定置信度 $1-\alpha$ 的置信区间为：

$$\overline{y}\pm\frac{s}{\sqrt{N-1}}t_{\frac{\alpha}{2}} \tag{9-45}$$

式中，$\overline{y}=\frac{1}{N}\sum_{i=1}^{N}y_i$ 是样本平均；$s=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\overline{y})^2}$ 是样本标准差；$t_{\frac{\alpha}{2}}$ 为自由度是 $N-1$ 的 t 分布上的 $\alpha/2$ 分位点。由绝对精度和相对精度的定义，样本平均的绝对精度是仿真次数和置信度的函数：

$$\delta(N,\alpha)=\frac{s}{\sqrt{N-1}}t_{\frac{\alpha}{2}} \tag{9-46}$$

相对精度就是

$$r(N,\alpha)=\frac{\delta(N,\alpha)}{|\overline{y}|} \tag{9-47}$$

为了得到要求的仿真精度，需要在仿真之前确定所需的最少仿真次数 N。然而，绝对精度和相对精度的计算需要知道样本 Y 的样本平均和样本标准差，一般情况下这在仿真进行之前是无法确定的（对于一些简单情况则是可以估算的，如利用式（9-44）等），因此最少的仿真次数并不能在仿真之前确定。所以，一种现实的办法是：首先设定一个基本的仿真运行次数 N。执行完毕后检验所得样本分布并计算仿真结果的精度，看是否达到要求，如果不满足要求，则继续执行下一次仿真并再次检验和计算仿真结果的精度，直到精度达到要求时停止仿真。这就是蒙特卡罗仿真次数的序贯算法，具体过程如下：

1）确定基本运行次数 N_0，最大运行次数 $N_{\max}$，要求的绝对精度 δ，相对精度 r 以及置信度 $1-\alpha$。

2）置仿真次数计数器 $n:=N_0$。执行蒙特卡罗仿真 N_0 次，得到试验样本 $\{y_1,y_2,\cdots,y_{N_0}\}$。

3）判断所得试验样本是否接近正态分布（可用前述的概率分布检验方法）。如果样本不是正态的，转 4）步：如果判断样本是接近正态分布的，那么计算：

$$A_n=\sum_{i=1}^{n}y_i\text{，}\quad B_n=\sum_{i=1}^{n}y_i^2 \tag{9-48}$$

4）再执行仿真一次，得到新的一个试验样本 y_{n+1}，并使仿真次数计数器加 1，$n:=n+1$，判断若 $n>N_{\max}$ 则认为算法失效并终止仿真，否则转 3）。

5）计算当前的样本均值、样本方差、绝对精度、相对精度，并与给定的精度要求进行比较。

$$\overline{y}(n)=\frac{A_n}{n} \tag{9-49}$$

$$s(n)=\sqrt{\frac{B_n-n[\overline{y}(n)]^2}{n}} \tag{9-50}$$

$$\delta(n,\alpha)=\frac{s}{\sqrt{n-1}}t_{\frac{\alpha}{2}} \tag{9-51}$$

$$r(n,\alpha)=\frac{\delta(n,\alpha)}{|\overline{y}(n)|} \tag{9-52}$$

如果精度满足要求，即 $0<\delta(n,\alpha)\leqslant\delta$ 且 $0<r(n,\alpha)\leqslant r$，或当前仿真次数 $n>N_{\max}$，则终止仿真，并输出计算结果的置信区间 $\overline{y}(n)\pm\delta(n,\alpha)$。否则，执行下一步。

6）执行仿真一次，得到新的一个试验样本 y_{n+1}，然后计算

$$A_{n+1}=A_n+y_{n+1}\text{，}\quad B_{n+1}=B_n+y_{n+1}^2 \tag{9-53}$$

并增加仿真计数器的值：

$$n:=n+1 \tag{9-54}$$

然后转 5）。

参 考 文 献



[1] 苏金明，王永利. MATLAB 7.0 实用指南：上册[M]. 北京：科学出版社，2000.
[2] 黄文梅，杨勇，熊桂林，等. 系统仿真分析与设计——MATLAB 语言工程应用[M]. 长沙：国防科学大学出版社，2001.
[3] 何强，何英. MATLAB 扩展编程[M]. 北京：清华大学出版社，2002.
[4] 曲永印. 电气电子变流技术[M]. 北京：冶金工业出版社，2002.
[5] 周渊深. 交直流调速系统与 MATLAB 仿真[M]. 北京：中国电力出版社，2003.
[6] 曾兴雯，刘乃安，孙献璞. 扩展频谱通信及其多址技术[M]. 西安：西安电子科技大学出版社，2004.
[7] 王忠礼，段慧达，高玉峰. MATLAB 应用技术——在电气工程与自动化专业中的应用[M]. 北京：清华大学出版社，2005.
[8] 王正林，王胜开，陈国顺. MATLAB/Simulink 与控制系统仿真[M]. 北京：电子工业出版社，2005.
[9] 周开利，康耀红. 神经网络模型及其 MATLAB 仿真程序设计[M]. 北京：清华大学出版社，2005.
[10] 吴旭光，杨惠珍，王新民. 计算机仿真技术[M]. 北京：化学工业出版社，2005.
[11] 刘白雁. 机电系统动态仿真——基于 MATLAB/Simulink[M]. 北京：机械工业出版社，2005.
[12] 张晓华. 系统建模与仿真[M]. 北京：清华大学出版社，2006.
[13] 唐向宏，岳恒立，郑雪峰. MATLAB 及在电子信息类课程中的应用[M]. 北京：电子工业出版社，2006.
[14] 钟麟，王峰. MATLAB 仿真技术与应用教程[M]. 北京：国防工业出版社，2007.
[15] 龚纯，王正木. MATLAB 语言常用算法程序集[M]. 北京：电子工业出版社，2008.
[16] 葛哲学. 精通 MATLAB[M]. 北京：电子工业出版社，2008.
[17] 劭玉斌. MATLAB/Simulink 通信系统建模与仿真实例分析[M]. 北京：清华大学出版社，2008.
[18] 姚俊，马松辉. Simulink 建模与仿真[M]. 西安：西安电子科技大学出版社，2008.
[19] 于浩洋，初红霞，王希凤. MATLAB 实用教程——控制系统仿真与应用[M]. 北京：化学工业出版社，2009.
[20] 谢仕宏. MATLAB R2008 控制系统动态仿真实例教程[M]. 北京：化学工业出版社，2009.